José Luis de la Vara, Gonzalo Génova, Jose María Álvarez-Rodríguez, Juan Llorens. (2017). An analysis of safety evidence management with the Structured Assurance Case Metamodel. Computer Standards & Interfaces, 50, pp. 179–198.

DOI: https://doi.org/10.1016/j.csi.2016.10.002

An analysis of safety evidence management with the Structured Assurance Case Metamodel

Jose Luis de la Vara*, Gonzalo Génova, Jose María Álvarez-Rodríguez, Juan Llorens

Computer Science Department, Carlos III University of Madrid, Avda. Universidad 30, 28911 Leganés, Madrid, Spain

A B S T R A C T

SACM (Structured Assurance Case Metamodel) is a standard for assurance case specification and exchange. It consists of an argumentation metamodel and an evidence metamodel for justifying that a system satisfies certain requirements. For assurance of safety critical systems, SACM can be used to manage safety evidence and to specify safety cases. The standard is a promising initiative towards harmonizing and improving system assurance practices, but its suitability for safety evidence management needs to be further studied. To this end, this paper studies how SACM 1.1 supports this activity according to requirements from industry and from prior work. We have analysed the notion of evidence in SACM, its evidence lifecycle, the classes and associations of the evidence metamodel, and the link of this metamodel with the argumentation one. As a result, we have identified several improvement opportunities and extension possibilities in SACM. The notions of evidence and evidence assertion should be clarified, the overlaps between metamodel elements should be reduced, and a wider support to the lifecycle of the artefacts used as safety evidence could be provided. Addressing these aspects will allow SACM to better fit safety evidence management needs and practices, especially beyond the scope of a safety case. The results and the conclusions drawn are especially valuable for practitioners interested in SACM adoption and vendors interested in developing tool support for SACM based safety evidence management.

Keywords: SACM Structured Assurance Case Metamodel Safety evidence Evidence management Safety assurance Safety certification

1. Introduction

SACM (Structured Assurance Case Metamodel; [1]) is an OMG (Object Management Group) standard that provides a common framework for assurance case development and information exchange. According to the SACM document [1], the standard has been created as a response to: (1) our growing reliance on systems whose dependability is critical for a wide variety of applications (transport, healthcare, energy, communications, banking, manufacturing, etc.); (2) the need for developing confidence in the quality of these systems; (3) the requirement of effectively justifying how dependability has been addressed and thus why we can rely on the systems, and; (4) the need for making system assurance more practical by enabling the automation of assurance related information management, and the meaningful representation and exchange of this information.

SACM defines 'assurance case' as a collection of auditable claims, arguments, and evidence created to support the contention that a defined system or service will satisfy certain requirements (e.g., safety and security requirements). Therefore, assurance cases are the basis for developing confidence in the quality of a system. In a structured argument, the relationship between the asserted claims and from the evidence to the claims is explicitly represented.

An assurance case could be targeted at, for instance, justifying that a system's timing constraints are satisfied (i.e., justifying why this claim is considered to be true). To this end, an argument on system verification and validation could be developed, e.g. because system stress testing has been performed and no system task deadline has been missed during system validation. Testing results and a testing verification report could be used as evidence. Assurance case management with SACM further aims to facilitate information exchange between the stakeholders of a system's lifecycle (system developer, component suppliers, regulators, operators, etc.) by explaining requirements satisfaction in a clear and defendable way.

The work related to SACM started in 2008, initial versions were published in 2010, and the approved 1.1 version has been publicly released in July 2015. The standard is divided into two main parts: the Argumentation Metamodel and the Evidence Metamodel. These metamodels were initially created independently and later combined to form SACM because they are complementary. Both research institutions (e.g., University of York) and companies (e.g., Adelard and

* Corresponding author.
E-mail addresses: jvara@inf.uc3m.es (J.L. de la Vara), gonzalo.genova@uc3m.es (G. Génova), josemaria.alvarez@uc3m.es (J.M. Álvarez-Rodríguez), juan.llorens@uc3m.es (J. Llorens).

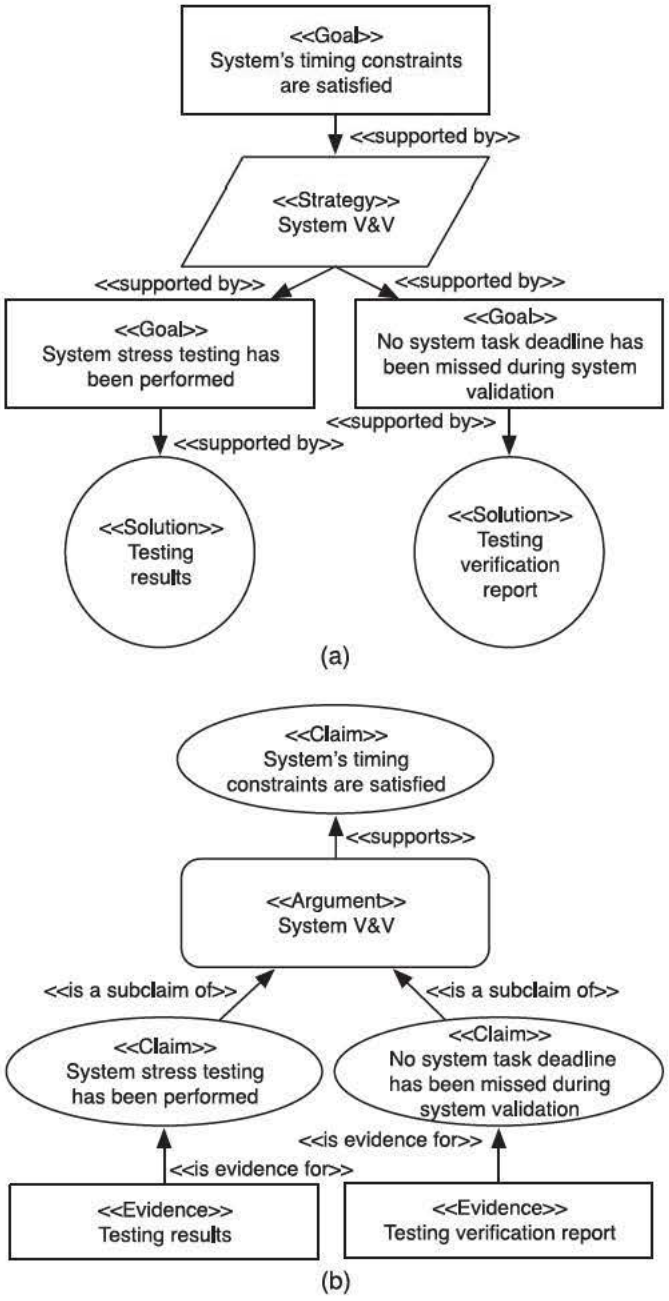

Fig. 1. Example of assurance cases with (a) GSN and (b) CAE.

Lockheed Martin) with wide experience in system assurance have contributed to the specification. The standard has been reviewed and approved, and is supported, by over 30 organizations, including Fujitsu, IBM, No Magic, PTC (formerly Atego), Rolls Royce, Sparx Systems, Thales, and Toyota. SACM is further aligned with GSN (Goal Structuring Notation; Fig. 1(a)) and CAE (Claims, Arguments and Evidence; Fig. 1(b)), two common techniques for assurance case specification in industry. A mapping between the elements of these techniques and SACM has been specified [1], and GSN and CAE diagrams can be represented with SACM concepts. Therefore, the standard covers assurance case constructs used in practice. SACM is used in practice both as a data exchange format (e.g. [2]) and as a data model for assurance information structuring (e.g. [3]).

Assurance cases and thus the notions of argumentation and assurance evidence have been used for decades in many application domains such as nuclear energy and railway. Their importance is also growing as a result of their recent introduction in domains such as automotive and healthcare [4]. An example of system assurance is safety certification, which can be roughly defined as a formal assurance by a third party (e.g., a certification authority) that a system fulfils its safety requirements and thus that the system does not pose undue risks to people, property, or the environment [5]. This formal assurance is typically based on the compliance with safety standards, such as IEC 61508 for electrical, electronic, and programmable electronic systems in a wide range of industries, DO 178C for avionics, the CENELEC standards for railway (e.g., EN 50128), and ISO 26262 for the automotive sector [6]. Demonstration of compliance with a specific standard involves gathering and providing convincing safety evidence: artefacts that contribute to developing confidence in the safe operation of a system and that can be used to show the fulfilment of the criteria of a safety standard. Examples of artefact types that can be used as safety evidence include safety analysis results, system specifications (e.g., requirements specifications), testing results, reviews, and source code [6]. For safety, an assurance case is usually referred to as safety case [5].

Safety evidence management can be divided into several general activities [6,7]. A company that develops safety critical products needs to decide upon the safety evidence to provide for a given system, collect the evidence, specify relationships in the body of evidence (i.e., between the artefacts used as evidence) and with other pieces of information (e.g., with an argument or a claim), assess the evidence, and perform evidence change impact analysis when necessary. Impact analysis can lead to the identification of issues that need to be addressed. For example, a piece of evidence can become invalid. Although focused on assurance cases, SACM provides concepts and relationships for modelling and thus dealing with safety evidence in general. SACM represents a promising initiative towards defining common, industrially agreed system assurance practices and improv ing them, including safety evidence management.

Safety evidence management can be very complex in industry. Practitioners can find difficulties in determining the information that has to be provided as evidence, in assessing the confidence in the evidence, in providing safety arguments, and in creating safety cases, among other issues [6,7]. Furthermore, inadequate safety evidence management can lead to certification risks [8]. These risks correspond to: (1) the risk of not being able to create a system that can be deemed safe; (2) the risk of not being able to provide a compelling safety case even though a system can be deemed safe, and; (3) the risk that a specific assessor or regulator will reject a safety case even though it is compelling to a general safety engineering audience. Therefore, it is crucial that a company uses adequate means for safety evidence management. Although SACM could be part of these means, it is currently very difficult to objectively state SACM suitability for safety evidence management because no study has analysed it in depth yet [9]. Providing a process for safety evidence management is out of the scope of SACM, but the classes and associations that the standard provides should allow a user to adequately specify the characteristics of the artefacts that are used as safety evidence, how they have evolved, and how they have been handled for a given assurance project.

This paper aims to determine how SACM 1.1 (hereafter referred to only as SACM for simplicity) supports safety evidence management. We have analysed the notion of evidence that SACM implies, the evidence lifecycle that it proposes, the 18 different class diagrams and over 100 classes of which the Evidence Metamodel consists, and the Argumentation Metamodel from a safety evidence management based perspective.

The analysis has been performed mainly from insights gained in OPENCOSS (http://www.opencoss project.eu/), a European industry academia project on safety certification for the automotive, avionics, and railway application domains. The OPENCOSS consortium consists of four research partners and 13 industry partners including system manufacturers, component suppliers, consultancy companies, safety assessors, certification authorities, and tool vendors. The project is also

supported by a large international advisory board with over 20 organizations from Asia, Europe, and North America. The insights have been gained in OPENCOSS from information about actual assurance practices at the OPENCOSS partners and the members of the advisory board, as well as at other companies that have participated in large surveys [6,7]. Practitioners have also validated the OPENCOSS results that correspond to turning these insights into conceptual frameworks and software tools. Therefore, we argue that the analysis of SACM can show the relevance of the standard to practice.

We have read the SACM specification and created SACM models (see examples in Appendix A) to determine what and how safety evidence information could be specified with the standard. This allows us to analyse whether SACM meets the safety evidence management needs that were identified in OPENCOSS (e.g., [10]) and to identify possible weaknesses. The needs are based on the requirements stated by the project's consortium and advisory board, and on requirements determined in large reviews of the state of the art [5] and practice [6,7]. The needs have been validated in OPENCOSS, including validation via three case studies in the automotive, avionics, and railway application domains, respectively. Nonetheless, this paper represents only our perspective on SACM and other people might not share them. We also use conceptual modelling principles [11] and model quality guidelines [12] in order to identify modelling needs and recommendations (e.g., avoidance of concept redundancy) that SACM might not be addressing adequately.

The analysis has resulted in the identification of elements that need to be clarified, might be inconsistent, might not be suitable, or could be included in SACM for safety evidence management. Based on the results from the analysis and on the current insights into SACM in the literature, we discuss improvement opportunities and extension possibilities. The results of the analysis can be very valuable for practitioners assessing SACM adoption, vendors that develop tool support for SACM and safety evidence management, researchers aiming to identify areas for further research on safety evidence management, and the people involved in the specification of the standard as they can identify possible improvements in future versions. In addition, and to the best of our knowledge, this paper contains the largest available SACM analysis.

The rest of the paper is organised as follows. Section 2 reviews related work. Section 3 analyses safety evidence management with SACM, whereas Section 4 synthesises and discusses the findings. Section 5 summarises our conclusions and future work. Finally, Appendix A presents examples of SACM models.

2. Related work

Assessing the quality and suitability of standard modelling languages has been a common strand of research work. BPMN (Business Process Model and Notation) [13] and UML (Unified Modeling Language) [14] are well known examples of these standards. The publications that have analysed them have found issues related to concept redundancy [15], language complexity [16], and inadequate concept representation [17], among others. This kind of study has aimed at identifying improvement opportunities and extension possibilities in the standards in order to facilitate their use. It is common that OMG standards are revised and new versions are released for enhancement.

We have previously shown that prior research has provided few insights into SACM [9]. SACM has been mostly referred to as an initiative that exists (e.g., [18]) or with which a correspondence could be established (e.g., [19]). Publications on safety evidence management [20], safety argumentation [4], safety compliance [19], and security assurance [21] have referred to SACM. We further concluded that detailed SACM analyses were necessary, including analyses of SACM suitability for specific purposes. Other aspects that could be addressed are the specification of further SACM usage examples, the use of SACM for assurance of various properties (e.g., safety and security), and the provision of more details about SACM relationships with other assurance approaches.

Some publications on safety evidence management have discussed SACM support for this activity and highlighted aspects that should be clarified. We have reported on our experience trying to adopt and adapt an earlier version of SACM for defining a safety certification metamodel [22]. We identified possible redundant classes in the evidence meta model, possible overlaps between the classes, and implementation decisions that might have been included. As presented in detail in Section 3, these kinds of issues remain in SACM 1.1. We also recommended carefully analysing SACM before deciding to use it as basis for another metamodel, and indicated that the notion of evidence in SACM might be unclear. Other publications have indicated the potential relationship of SACM with their proposals for safety evidence lifecycle [23], for characterising safety evidence assessment [24], and for characterising safety evidence in general [25,26]. Nair et al. [5] indicate that SACM does not provide a thorough and sufficiently detailed analysis of the possible evidence types to provide for safety certification and of how to structure and assess evidence. Li et al. [27] consider that SACM lacks support for the evidence collection process. As positive aspects, it has been acknowledged that SACM provides many attributes for describing evidence [28] and rich support for the specification of evidence provenance [29].

Although these publications have provided valuable insights into SACM, their analyses have been partial. To the best of our knowledge, no publication up to now has studied in depth how safety evidence management can be addressed with all the classes and associations of SACM.

Regarding prior work on safety evidence management, recent publications have presented large studies on the state of the art [5] and on the state of the practice [6,7]. It is also easy to find deliverables in research projects that have reviewed the literature and industrial practices (e.g., [26]). Prior work has proposed non standard models for managing safety evidence [19]. Some are generic (e.g., [20]) and others have been created for safety evidence management according to specific safety standards, e.g. IEC 61508 [30].

Finally, there exist other standardization efforts for assurance cases in addition to SACM. There is a GSN community standard [31], the Open Group has developed the Dependability through Assuredness Framework [32], IEC 15026 2 [33] defines assurance case concepts, and IEC 62741 addresses dependability case management [34]. Several domain specific system assurance standards (e.g., ISO 26262 in automotive) also provide indications about the intent and expected content of an assurance case [4], mainly for safety critical systems. Nonetheless, SACM is currently the only standard metamodel for creating and exchanging structured assurance cases. Other metamodels for assurance cases can also be found in the literature (e.g., [35]), but they are not standards and have been developed by a lower number of parties.

3. SACM support for safety evidence management

This section analyses how SACM supports the management of safety evidence (safety analysis results, system specifications, testing results, etc.), focusing on the identification of possible issues that could hinder this activity. As explained in the introduction, the analysis is based on requirements for safety evidence management from industry and from prior work, and on modelling and model quality principles. We use the hazard log and software verification results as main running examples for presenting the analysis. A hazard log can be defined as the document in which all the safety management activities, hazards identified, decisions made, and solutions adopted, are recorded or referenced [36]. Software verification results indicate the procedures that passed or failed for each software review, analysis, and test, including coverage analyses and traceability analyses [37]. Both types

of artefacts are commonly managed in most safety critical domains and used as safety evidence [6]. For example, a hazard log could be used as evidence of closure of all safety requirements by means of e.g. tests and analyses. Appendix A contains SACM models of the examples used in this section.

SACM aims to provide a modelling framework that allows users to express and exchange their argument structures and the associated evidence information. The standard contains those elements that its authors regard as fundamental for these tasks, and the elements are aligned with current assurance case specification practice, e.g. with GSN and with CAE. SACM is divided into an Evidence Metamodel and an Argumentation Metamodel.

The Evidence Metamodel defines a catalogue of elements for constructing and interchanging precise statements involved in evi dence related efforts. This metamodel aims to: (a) identify the main factors that determine the evidence collection process; (b) identify the main factors that determine the evaluation of evidence; (c) identify and define the elements of evidence; and, (d) define a common interchange format to facilitate the exchange of information between different software assurance tools and services.

The Argumentation Metamodel defines the catalogue of elements for constructing and interchanging structured statements describing argumentations. SACM defines 'argument' as a body of information presented with the intention to establish one or more claims through the presentation of related supporting claims, evidence, and contextual information. A structured argument is a particular kind of argument where the relationships between the asserted claims, and from the evidence to the claims, are explicitly represented. Evidence and thus evidence management are essential aspects in argumentation.

SACM aims to allow a user to independently use the Argumentation and Evidence Metamodels. For example, a user should be able to create Argument Models without using the Evidence Metamodel. In this sense, SACM distinguishes between evidence specific information, which does not depend on the argument structure in which evidence is used (e.g., the creator of a document), and argumentation based evidence information (e.g., regarding evidence support to a given argument claim; aka subject claim). The first type of information can be reused in different arguments. SACM explicitly indicates that Evidence Models can be used in support of multiple assurance cases.

The Evidence Metamodel has four main logical parts: (1) evidence items (e.g., the software verification results) part, which defines the physical evidence (e.g., the document that reports software verification results); (2) formal elements part, which defines the logical assertions, provided in the form of individual propositions (e.g., the definition of 'software verification results' as a formal object for making assertions about it in an assurance case); (3) evidence assertions part, which defines various statements that can be made about the evidence items (e.g., for indicating that the custodian of the software verification results is a given person), and; (4) administration part, which can be used to organise individual evidence items and evaluations into a package that becomes a unit of exchange (e.g., for specifying that the software verification results are part of a larger evidence container called 'software artefacts').

The analysis performed on SACM has been divided into nine areas: notion of evidence, evidence lifecycle, evidence elements, exhibit properties, formal statements, evidence properties, evidence evalua tion, administration, and argumentation. SACM class diagrams, classes, attributes, associations, enumerations, and enumeration lit erals are highlighted in italics hereafter. Further details about SACM elements (e.g., their definition and information about their semantics and constraints) can be found in the SACM specification [1].

3.1. Notion of evidence

In general, evidence can be defined as the available body of facts or information indicating whether a belief or proposition is true or valid [38]. In SACM, *Evidence* is defined as information or objective artefacts being offered in support of one or more claims, whereas *Evidence Item* is defined as a unique element of the body of evidence, such as an exhibit, a claim, or other element of meaning associated with an exhibit.

Many experts have agreed upon the need for evidence based safety demonstration, based on the 'guilty until proven innocent' argument (e.g., [39]): a system should not be deemed safe unless safety is demonstrated, and safety evidence must be provided. It is true that a system supplier might develop a safe system despite not being able to demonstrate it, but this is not acceptable for critical applications. On the other hand, safety of a real, large system cannot be really shown, and some degree of uncertainty will always exist. For example, the inexistence of residual errors cannot be demonstrated. Therefore, and as introduced above, safety evidence can be defined as artefacts that contribute to developing confidence in the safe operation of a system and that can be used to show the fulfilment of the criteria of a safety standard [5]. For example, a hazard log cannot be used as proof of a complete absence of hazards for a system, but can help a certifier develop confidence in its safe operation. Many experts use the notion 'system X is acceptably safe' (e.g., [4]). Acceptability and thus confidence in safety will depend on a system's purpose, usage, and application domain.

The main difference between the definition in the previous para graph and SACM definitions is that we do not think that information per se (e.g., a statement), without being provided as part of some tangible element such as a document or in an electronic file such as a system diagram created with a modelling tool, should be regarded as safety evidence. Safety evidence should only correspond to existing, available artefacts, which can be used as evidence, e.g. a hazard log. Some SACM concepts are classified as evidence items but in our opinion they should not be regarded as evidence. For example, the overall definition of *Evidence Item* refers to a claim as one of the elements that is an evidence item, and *Formal Assertion* (see Sections 3.3 and 3.5) specialises *Evidence Item*. We think that a claim or SACM formal assertion (e.g., "the hazard log shows the closure of all safety requirements") should not be regarded as safety evidence. Although they are information, and more concretely they can be assurance information, they are not artefacts. A safety claim can contribute to developing confidence in the safe operation of a system, but its validity and truth should be challenged unless the claim is supported by evidence. Basically, a claim itself, in isolation, barely contributes to safety demonstration. Stating that all the reviews of a software system have been passed, without providing evidence in the form of software verification results, is not safety evidence. Classifying a claim as an SACM evidence item can lead to a wrong interpretation of what safety evidence is, and thus to weak safety assurance, especially considering all the characteristics that SACM defines for *Evidence Item* (see e.g. Section 3.4). We think that information should be regarded as evidence only when it is available in some artefact, which could be used as evidence.

We have previously discussed [20,22] that an artefact is not safety evidence in itself, in isolation, but that an artefact can be used as evidence of some safety claim. For example, a hazard log is an artefact that can be used as evidence of the claim 'system hazards have been identified'. In other words, an artefact plays a role as evidence when associated to a claim. This is shown in Fig. 2, which we have created

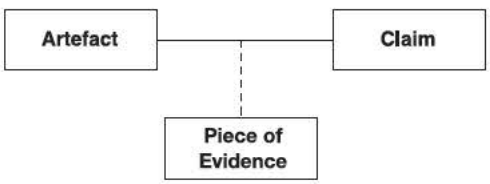

Fig. 2. Overall notion of safety evidence.

and makes an explicit difference between an artefact and a piece of evidence. As a result of its use as safety evidence, an artefact can have new properties and relations. For example, the use of an artefact as evidence can support a claim, with a given level of confidence. The most similar notions to piece of evidence in SACM are the *Asserted Evidence* and *Asserted Counter Evidence* classes of the Argumentation Metamodel (see Section 3.9), but not *Evidence Item*. However, the Argumentation Metamodel allows the use of *Evidence Element* (see Section 3.3), which generalises many types of information that are not artefacts, as evidence or counter evidence of a claim. This is a too broad usage in our opinion.

We further think that confusion can easily arise in SACM usage because of specifying specialisation relationships such as 'a document is an evidence item' (see Section 3.3). As indicated in conceptual modelling principles [11], this specialisation implies that a document has the same attributes as an evidence item. However, and as discussed above, a document can exist without being used as evidence. Therefore, we advocate the perspective of 'a document is used as an evidence item'. Although an introductory passage of SACM indicates that a document becomes evidence only when it is claimed to provide evidentiary support to a certain subject claim ([1]; Section 7.6, page 24), this notion is not represented in the Evidence Metamodel.

In summary, and as further discussed below, we think that the unclear definition and use of *Evidence Item* in SACM can lead to ambiguity and misinterpretation of what safety evidence is and thus to inadequate safety evidence management.

3.2. Evidence lifecycle

Fig. 3 shows the lifecycle proposed in SACM for *Evidence Item*.[1] Although the lifecycle is non normative, it provides a view of how SACM considers that an evidence item can evolve. The lifecycle consists of a single state after an evidence item has been created, acquired, or derived, and before it is revoked. During its lifecycle, an evidence item can be transferred, evaluated, or re evaluated. Although the usefulness of a lifecycle with a single state might be debatable, the proposed lifecycle at least explicitly indicates the possible events to create an evidence item and the possible events until it is revoked.

There are some aspects of evidence lifecycle that do not seem to have been considered in SACM, or at least that are not clearly supported in the proposed lifecycle. Firstly, the safety evidence to provide for a system must be determined at the beginning of a project, and usually in agreement with e.g. a certification authority. This is the focus of e.g. Falessi et al. [40]. This publication indicates that the safety evidence to provide for a system must be specified and agreed between a system supplier and a certification authority. Several iterations might be necessary before reaching the agreement.

Secondly, safety evidence evolution and change impact analysis could be better supported in SACM. In the scope of OPENCOSS [10,41], we have drawn the conclusion that an artefact used as safety evidence can have several states in a project. In a general and abstract way, and depending on the constraints specified for a project, the results of a test case might be valid as safety evidence, might be invalid, might need to be validated (i.e., it might have to be decided whether it is valid), or might need to be revoked. Test case results might be revoked if e.g. the tested source code changes, thus the test case should be re executed. Other evidence lifecycles can also be found in the literature [26].

Thirdly, we consider that the lifecycle proposed for evidence item might be inaccurate because all the elements that specialise *Evidence Item* (see Section 3.3) do not seem to follow it. For example, it does not seem suitable to us to model that an *Object* (e.g., the concept 'hazard'; see Section 3.5) has been evaluated.

[1] All the figures of SACM elements and class diagrams correspond to excerpts from the SACM specification [1].

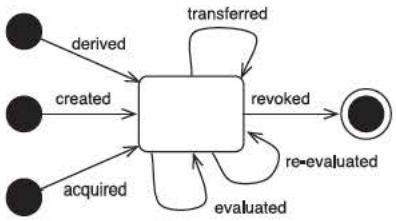

Fig. 3. Evidence item lifecycle in SACM.

Nonetheless, we acknowledge that evidence lifecycle, especially in relation to safety evidence evolution and change impact analysis, is an area that might require further research. An open question is whether a 'standard' evidence lifecycle can really be defined. We have observed that companies have their own lifecycles (e.g., with a state called 'frozen' to indicate that no changes are allowed to a given artefact), which might not be compatible with others. A possible solution could be to specify a generic, more abstract lifecycle. However, the gap between this lifecycle and the one used in a given company could be too big, hindering the usefulness of the standardised evidence lifecycle in practice. The challenge grows if the differences in safety assurance and certification practices among application domains and among certifica tion authorities are also considered [6].

3.3. Evidence elements

The Evidence Elements section of the Evidence Metamodel consists of the *Evidence Elements* and *Evidence Assertions* class diagrams. This section defines the key concepts of the Metamodel.

In the *Evidence Elements* class diagram (Fig. 4), *Evidence Item* is defined as an abstract class that represents objects that are collected as evidence. For example, a hazard log would be a document, and an evidence group might be created for compiling this log and other artefacts related to safety analysis results [5], such as documents with fault tree analysis, failure mode and effect analysis, etc.

In our opinion, and in line with the explanation in Section 3.1, some classes that specialise *Evidence Item* should not be regarded as safety evidence. We consider that *Exhibit*, *Document*, *Record*, and *Evidence Group* are suitable, but *Formal Object* and *Formal Assertion* are not. We suggest that *Formal Object* and *Formal Assertion* are removed from the specialisation hierarchy shown in Fig. 4.

The distinction between *Record* and *Document* is also strange to us because we understand that *Record* can have the same attributes and associations as *Document* and *Exhibit* (see Section 3.4). The only difference between the classes is that *Exhibit* is an object itself and *Record* overall represents information about a happening. For exam ple, the minutes of a meeting is a record for SACM, whereas a design specification is a document (exhibit). However, the need for this distinction is not clear in relation to their use as safety evidence. One could also easily argue that a design specification represents the fact of having designed a system, which matches *Record*. In essence, both *Exhibit* and *Record* seem to match our definition of safety evidence. It is also confusing that the description of *Document* in SACM refers to a recording as something that is a document, and that the description of *Record* indicates that records represent exhibits. Indeed, we consider that the definition, description, and semantics of *Exhibit*, *Document*, and *Record* should be improved for clarity and thus facilitating the correct use of these classes.

Another major issue that we encounter in SACM is the granularity that it implies for the artefacts managed as safety evidence. Many artefacts that can be used as safety evidence do not fit the granularity *Document* and *Record*, but would be parts of, for instance, a document and would not constitute documents themselves. For example, a document such as a requirements specification could contain indivi dual safety requirements, which correspond to safety evidence. These

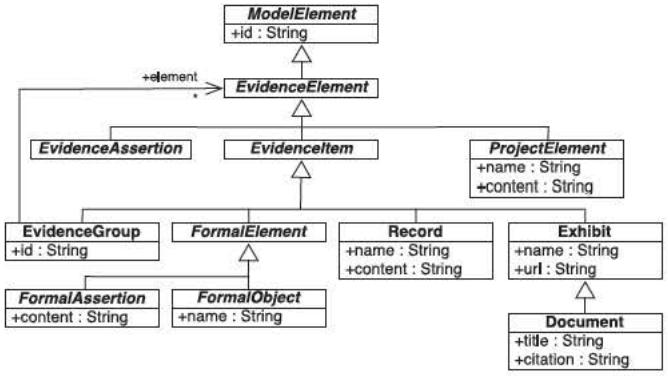

Fig. 4. Evidence elements class diagram.

requirements can be referred to individually in a hazard log: for each hazard, the corresponding safety requirement that mitigates or avoids the hazard must be determined.

It would also be beneficial that SACM provided guidance on when to use *Evidence Group* and when to use *Evidence Container* (see Section 3.8), and constraints upon the evidence elements for which provenance, timing, custody, and event information can be specified. For the latter, and as for other classes of the Evidence Metamodel, the current scope of *Evidence Element* regarding the specification of such information is too broad. More event types are defined (see Section 3.6) than what it is probably necessary for the management of e.g. the concept 'hazard' (*Object*).

Evidence Assertion (Fig. 5) corresponds to statements about evidence items. This includes: essential properties of evidence items; properties of documents; statements about the custody, provenance, and timing of an evidence item; attributes of the evidentiary support; interpretation of the evidence; nature of the evidentiary support; observations; resolutions, and; standard of proof for evidence. *Evidence Attribute*, *Evidence Property*, and *Evidence Evaluation* are further presented in SACM in other class diagrams.

It is strange to us that attributes and properties are classified as *Evidence Assertion*. For example, an evidence assertion would be used to specify that the results of a test case are part of software verification results, or to specify the version of a hazard log. This way of specifying characteristics is very different to other metamodels, including OMG ones, in which attributes and properties are included in classes instead of being regarded as statements about the classes. We think that this SACM characteristic can make it difficult both to understand concep tually and to align with other metamodels. We also discuss below that some evidence attributes and properties should probably be evidence evaluations.

SACM proposes a lifecycle for *Evidence Assertion* that, as for *Evidence Item* (Section 3.2), consists of a single state. Once an evidence assertion is created, it can be evaluated or re evaluated before it is revoked. SACM has not taken into account the possibility that an evidence assertion can be regarded as valid or as invalid during its lifecycle. Further operations on an evidence assertion (e.g., the specification of its support to some claim) are not represented. As with *Evidence Item*, we consider that the lifecycle proposed for *Evidence Assertion* is too broad because some of the classes that specialise this class do not follow the lifecycle. This is an important conceptual error in our opinion. For example, it does not seem suitable to us to model that a *Timing Property* of a piece of safety evidence such as *At Time* (Section 3.6) has been evaluated.

We also think that different names to some of the classes that specialise *Evidence Assertion* would make their intent and scope much clearer. For example, the difference between *Evidence Attribute* and *Evidence Property* can be difficult to ascertain. These classes are described below. In fact, we think that the former should be called 'Evidence Evaluation Attribute', and the latter 'Evidence Element Property'. Classes that refer to aspects of an artefact that are indepen dent of its use as evidence (e.g., its completeness; see Section 3.4) should probably not start with 'Evidence'. In addition, since evidence assertions can be used to model relations with subject claims, then evidence assertions of the Evidence Metamodel overlap with the Argumentation Metamodel. Overlaps between and redundancy of concepts, as with *Evidence Assertion* and *Claim*, are negative for the quality of a model or metamodel [12]. This is different to referring in a class diagram to a class that is introduced in another class diagram. Obviously, the interface between two parts of a model or metamodel has to be used in both places; this is not the kind of redundancy that we criticise. Finally, constraints should be specified to indicate which evidence elements can actually have each *Evidence Property*, to specify their consistency needs, and thus to correctly use the metamodel.

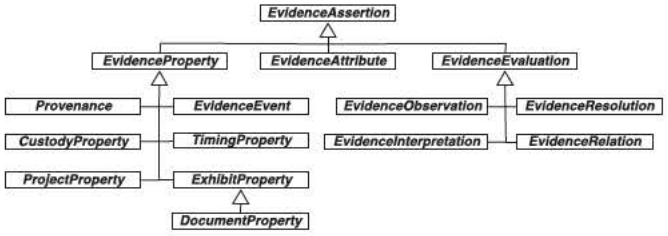

Fig. 5. Evidence assertions class diagram.

3.4. Exhibit Properties

The Exhibit Properties Section of the Evidence Metamodel consists of the *Exhibit Properties* and *Document Properties* class diagrams. This section defines elements that allow constructing statements about the fundamental properties of exhibits and documents.

With *Exhibit Property* (Fig. 6), someone could specify that a hazard log entry (exhibit) is part of a hazard log (another exhibit), a hazard log can be based on a safety plan (evidence item), and a hazard log can have an electronic source in the form of a spreadsheet.

An unclear aspect of the class hierarchy is that an exhibit can be based on an evidence item, which can correspond to some concepts that do not match our notion of safety evidence. For example, it can be specified that an exhibit (e.g., a hazard log) is based on, and thus has been generated from, a formal object (e.g., the concept 'hazard'). The use of *Is Based On* should probably only be allowed between exhibits. *Is Part Of* can only be specified in relation to another exhibit, which seems more suitable to us. The multiplicity of the associations of *Is Based On* and *Is Part Of* with *Evidence Item* and *Exhibit*, respectively, is '1', and *Is Based On* and *Is Part Of* do not seem to need further attributes or associations. Therefore, based on conceptual modelling principles [11], *Is Based On* and *Is Part Of* are unnecessary (as classes) because (1) instances of them can only exist when associated to other classes and (2) the classes do not have other properties (e.g., emerging properties as a result of creating an association, as in the 'Piece of Evidence' class in Fig. 2). They should be modelled as associations of *Exhibit* with *Evidence Item* and *Exhibit*.

SACM should also provide mechanisms for modelling further types of relationships and properties for exhibits, and allow the specification of more than one *Has Electronic Source*. If someone thinks of for example an architecture model as an exhibit, such a model could be specified in several electronic sources (files) representing different

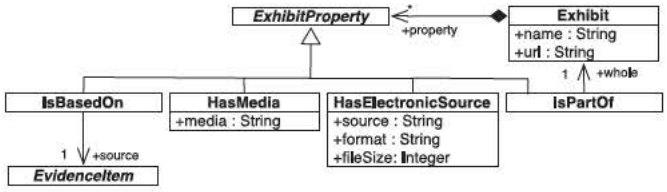

Fig. 6. Exhibit properties class diagram.

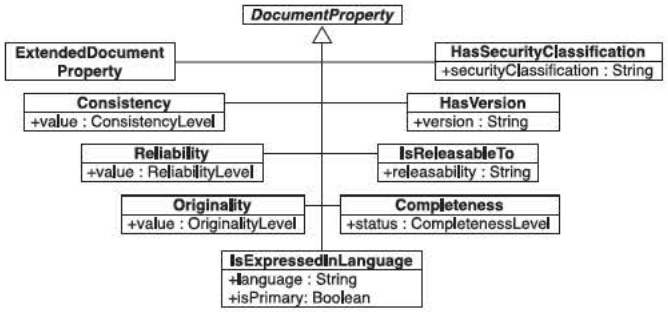

Fig. 7. Document properties class diagram.

architectural views. The support to the specification of further types of relationships is especially important because of the major role of traceability in the assurance and certification of safety critical systems [42].

The *Document Properties* class diagram (Fig. 7) defines several possible characteristics of a document. *Document Property* specialises *Exhibit Property* (see Fig. 5). For some of these characteristics, a value in the form of a level can be specified from a predefined set of levels in an enumeration. For example, *Consistency* can be *Unknown*, *Informal*, *Semiformal*, or *Formal*. *Extended Document Property* can be used for defining further characteristics (see e.g. [24] for other possible characteristics).

The unclear distinction between *Document* and *Record* becomes even more evident when considering document properties. We consider that these properties could be equally specified for a record (e.g., the minutes of a meeting).

Another aspect that has caught our attention is the different nature of the document properties regarding how to specify them. More concretely, we think that some properties of a document (e.g. a design specification) can be regarded as objective (e.g., *Has Version*), whereas others are subject to interpretation (e.g., *Reliability*, from unreliable to completely reliable). For the latter, we consider that it would probably be better to regard the properties as evaluations. They require a judgement, and the assigned value might have to be justified. This could even be done by means of an argument, in which someone could justify, for instance, why a document has been classified as *Fairly Reliable*.

In addition, the use of the *Has Version* class does not seem to be suitable in SACM. A document (e.g., the hazard log) could have several versions, and each version could have different properties and be used for different purposes. For example, a given version of a document might be *Incomplete*, but another *Final*. Even different evidence assertions could be made for different versions of a same document. However, SACM does not adequately support the specification of this kind of information, and forces the creation of a new document for each new version if e.g. different properties have to be specified for different versions. We consider this to be an important weakness in SACM, and it can lead to issues when aiming to follow a document's lifecycle. For example, it can be hard to track the version of a hazard log used in a specific system lifecycle activity.

Constraints should also be explicitly added regarding the possibility of specifying document properties only for documents, and how many. And finally, we feel that most of the classes of the Exhibit Properties Section could be turned into attributes of the other classes (e.g., *Completeness* could be an attribute of *Document*), or even should be according to conceptual modelling principles [11]. The information for the instances of the classes can only be specified in the scope of, for instance, a given document, and we understand that such information cannot be reused among different documents. Nonetheless, the way SACM has been structured could result from an implementation decision. The authors of SACM might have preferred that the users explicitly add attributes to documents. Although we think that this is conceptually strange and could even be regarded as incorrect, it is valid from an implementation perspective.

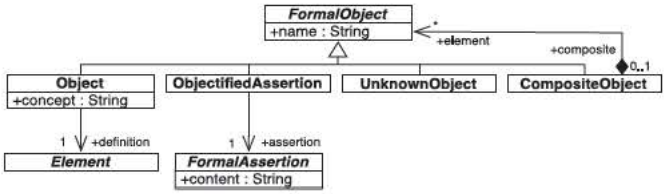

Fig. 8. Formal objects class diagram.

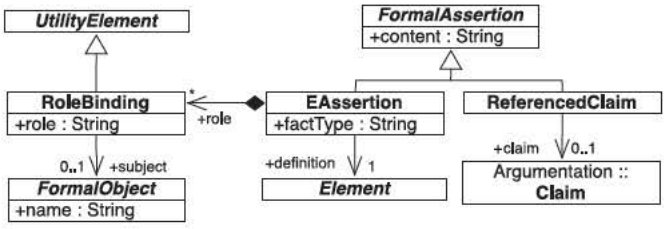

Fig. 9. Formal assertions class diagram.

3.5. Formal statements

The Formal Statements section of the Evidence Metamodel provides means for representing the elements of meaning involved in the processes of interpretation and evaluation of evidence required for precisely representing assertions. This section consists of the *Formal Objects* (Fig. 8) and *Formal Assertions* (Fig. 9) class diagrams, and provides a way to specify and structure the concepts and terms used in an SACM Model, e.g. the concept and term 'hazard log' of our running example. The section is aligned with and relates to SBVR (Semantics of Business Vocabulary and Rules) [43].

We have not found any major issue regarding how formal statements can be specified (i.e., in their syntax and structure) beyond the facts that formal assertions might refer to subject claims, thus there is an overlap with the Argumentation Metamodel, and that a clearer explanation of the difference between *Formal Assertion* and *Evidence Assertion* would be helpful. Nonetheless, we have some concerns regarding the purpose, use, and semantics of formal statements.

Firstly, it can be hard to understand why the *Formal Objects* and *Formal Assertions* class diagrams have been specified when SBVR is available. We wonder if a subset of SBVR could have been used. Secondly, the use of formal statements could easily increase the complexity of SACM application if, for instance, someone decided to specify all or most of the concepts and terms in an SACM Model with formal statements. Therefore, we think that the purpose of the formal statements should be clarified, as well as when they should be used and how.

Finally, this section of SACM includes references to other sections and descriptions that can be hard to understand. For example, SACM text says: "Usually a Formal Object *corresponds to* an Exhibit where the Exhibit element emphasizes the physical object (an instance of the SBVR 'Thing' concept) while a Formal Object emphasizes the associated element of meaning (an instance of the SBVR 'Meaning' concept)". Furthermore, circular references are specified in the *Formal Assertions* class diagram. This diagram refers to *Claim* from the Argumentation Metamodel, which also refers to the Evidence Metamodel. Circular references between the class diagrams of a metamodel can cause problems when creating or analysing a model [44–46].

In summary, why and how to use formal assertions, and more specifically for safety evidence management, is not completely clear to us. Further usage examples should be provided. We also suggest that the Formal Assertions Section should be a separate metamodel to

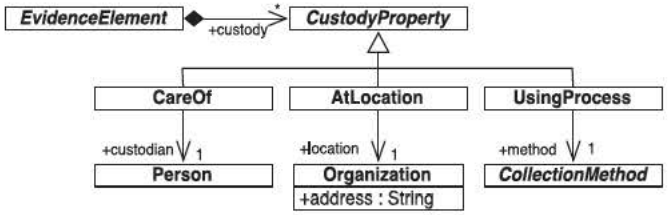

Fig. 10. Custody class diagram.

which the Argumentation Metamodel and the Evidence Metamodel refer.

3.6. Evidence properties

The Evidence Properties section of the Evidence Metamodel consists of the *Custody*, *Evidence Events*, *Provenance*, and *Timing* class diagrams. All the root classes of these class diagrams (named as the class diagram, but in singular) are part of *Evidence Element* (composition association).

Someone could specify with *Custody Property* (Fig. 10) that a given safety assurance manager is the custodian of an evidence element such as a hazard log, and that the hazard log has been created according to the indications of a given reference technique that indicates its expected structure and content (collection method).

An aspect that we do not understand in this class diagram is that the multiplicity of the three associations is '1'. The three classes also do not seem to have any specific relevant attribute, thus they might be unnecessary based on conceptual modelling principles [11]. The same information could be specified if the classes were removed and the associations were established between *Evidence Element* and *Person*, *Organization*, and *Collection Method*. The only association of *Care Of*, *At Location*, and *Using Process* is in the *Evidence Events* class diagram, but also in this case the same information could be specified without the classes. SACM should also include constraints regarding the use of custody properties (e.g., how many an evidence element could have).

The attributes and associations of the classes that specialise *Evidence Event* (Fig. 11) are specified by means of tables that list their properties. This is different to the rest of SACM classes, thus we consider that a reader can get confused. Furthermore, the tables do not explicitly show what type of attribute or association each property corresponds to. This information should be included. The events seem to have associations with unnecessary classes. This should be revised.

Another issue is that the *Evidence Events* class diagram is not consistent with the *Evidence Item* and the *Evidence Assertion* life cycles, as the sets of events do not coincide. SACM should be clearer about which evidence events could be specified for the different evidence elements, and provide constrains for *Evidence Event* use (e.g., regarding their number). The description of the evidence events should also be revised because we have identified some inconsistencies, and more concretely regarding their properties. For example, we do not understand why *Owned By* can be specified for all evidence event subtypes but *Is Modified By*.

Regarding *Provenance* (Fig. 12), the source and owner of software verification results could be a verification manager, a tester could be among its executors, and a manager might approve it (supervisor). Again, and as argued for the specialisations of *Custody Property*, these

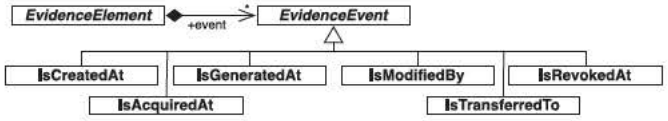

Fig. 11. Evidence event class diagram.

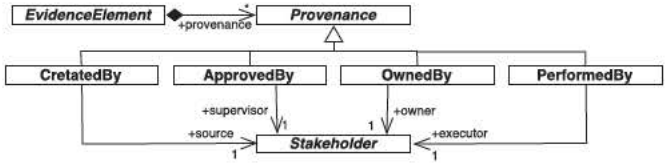

Fig. 12. Provenance class diagram.

four classes might not be necessary. The same information could be specified without them, directly associating *Evidence Element* four times with *Stakeholder*. *Provenance* also seems to overlap with *Evidence Event*, and constraints should be included in SACM regarding what type of provenance apply to the different evidence elements and regarding how many provenance properties can be specified for a given evidence element. For example, we understand that the property *Performed By* is not suitable for *Object* (e.g. the term 'hazard'), and that an object should not have more than one *Created By*.

Analysing the *Custody* and *Provenance* class diagrams in more depth, both might be unnecessary. The same information of the alternative associations proposed above could be specified without the *Custody Property* and *Provenance* classes. Instead of creating associations with these two classes, the associations could be with *Evidence Element* and with *Evidence Event*. The only reason that we can find for keeping the *Custody Property* and *Provenance* classes is that someone wanted to specify e.g. when provenance started ('the hazard log is owned by the safety manager since June 10, 2016'). However, this can be specified with evidence events. Furthermore, the *Custody*, *Evidence Events*, and *Provenance* class diagrams lack constraints regarding the fact that some properties must only occur once. For example, a hazard log is created only once, and can be only at one location.

We do not see the need for the *Timing* class diagram (Fig. 13). According to conceptual modelling principles [11], its information should be represented with attributes (e.g., of *Evidence Event*). It does not seem suitable to us to specify a timing property on its own, it has to be specified only when used in the scope of other classes, and a timing property cannot relate to several instances of other classes. In addition, it is not clear to us what *Effective Time* of *Evidence Event* means. *Effective Time* is an abstract class, thus one should use *Start Time* or *End Time* for *Evidence Event*, and referring to the start or end times of an event is not logical in our opinion. An event such as the creation of an artefact simply happens at a given moment and has no duration, so it is enough with the information of *At Time* in *Evidence Event*. This is also how events are understood and used in other modelling languages, e.g. BPMN and UML. In addition, constraints should be included necessary in SACM to avoid that e.g. two instances of *At Time* are assigned to an event.

Finally, it is clear to us that not all SACM evidence elements (e.g., *Organization*) need all the above properties (e.g., *Performed By*). A different class structure or constraints must be specified in SACM. This would very likely make the metamodel easier to understand and thus use.

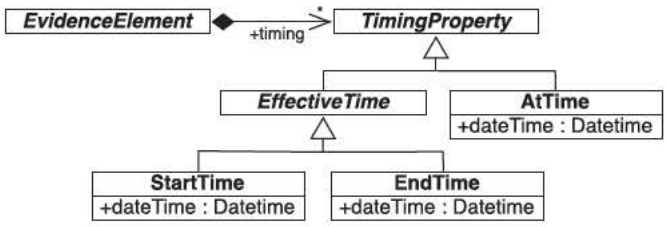

Fig. 13. Timing class diagram.

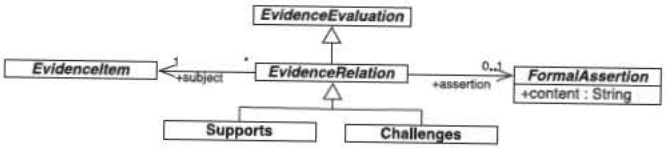

Fig. 14. Evidence relation class diagram.

3.7. Evidence evaluation

The Evidence Evaluation Section of the Evidence Metamodel consists of the *Evidence Relations*, *Evidence Attributes*, *Evidence Interpretation*, *Evidence Observations*, and *Evidence Resolutions* class diagrams. A general characteristic of this section is that it overlaps with argumentation aspects. SACM explicitly indicates that evidence evaluations provide information about the relationships of evidence items with subject claims (i.e., claims in an argument structure), and we consider that most of the content of this section should be, or already is, part of the Argumentation Metamodel. Much information about evidence evaluations seems to correspond to confidence arguments [48], which aim to justify the adequacy of a given argument structure. All argumentation related aspects should be moved to the Argumentation Metamodel. In addition, most evidence evaluations are not reusable between assurance cases because they depend on subject claims, i.e. evidence evaluations in the scope of a subject claim (e.g., the support of the hazard log to a given safety claim) cannot be reused for another claim. This contradicts SACM purpose regarding the use of Evidence Models in support of multiple assurance cases.

The *Evidence Relations* class diagram (Fig. 14) supports the specification of information such as "the evidence item 'hazard log' challenges the SACM formal assertion 'all the identified hazards have been mitigated or avoided'". Although this kind of information is necessary for safety evidence management, the allowed use of the *Supports* and *Challenges* classes is too broad. For example, we do not see why someone would want to specify that an SACM formal object, in e.g. the form of a concept such as 'hazard', supports a formal assertion (e.g., 'the hazard log shows the closure of all safety requirements'). The kind of assertions that can be made with *Evidence Relation* only seems suitable to us in the scope of an argument. We further discuss below (Section 3.9) the possible overlap between the Argumentation and Evidence Metamodels of SACM. It is also not clear to us why *Evidence Relation* does not need to be always associated with *Formal Assertion*. It does not seem logical to us that someone specifies *Supports* or *Challenges* for *Evidence Item* without referring to something.

The classes that specialise *Evidence Attribute* (Fig. 15) have an attribute to specify their value, and in most of the cases according to an enumeration. *Confidence* can for instance be *Unknown*, *Reported As Uncertain*, *Reported As Plausible*, or *Reported As Fact*. For the example in the previous paragraph regarding hazard mitigation or avoidance, the confidence in the assertion could be *Reported As Fact* (because the hazard log shows that some hazards have not been mitigated or avoided yet). Custom evidence attributes can be specified by means of *Extended Evidence Attribute*.

We find four issues in the *Evidence Attributes* class diagram. First, the possibility of assigning evidence attributes to evidence evaluations is too broad. SACM allows the specification of evidence attributes that do not seem suitable to us, such as "the strength of 'Requirement R is a requirement' is 89". We also do not see when some evidence interpretation might need an evidence attribute. Second, SACM proposes several enumerations for rating the attributes, but provides little information about what the literals of the enumerations mean or how to decide upon the levels. Without such information, inconsistent use seems likely. For example, we wonder what 'the significance is medium Low' would mean. It can also even be difficult to distinguish between certain evidence attributes (e.g., *Relevance* and *Significance*). Therefore, it is likely that different people end up using *Evidence Attribute* in inconsistent, different ways. Third, we think that the specialisations of *Evidence Attribute* might not be necessary, except *Extended Evidence Attribute*. The same information could be specified by means of attributes in *Evidence Evaluation*, without having to define new classes. In addition, and as discussed above for *Timing Property*, it is not logical to declare an evidence attribute unless it is associated to an evidence evaluation. Nonetheless, this could simply be an implementation decision in SACM. Another reason could be that SACM aims to allow a user to simultaneously specify, for instance, several levels of confidence for a same evidence evaluation. However, this is not explained and can even be considered illogical. Finally, SACM should specify constraints for the adequate use of *Evidence Attribute* with other evidence elements. For example, *Relevance* should only be specified when *Supports* is indirect.

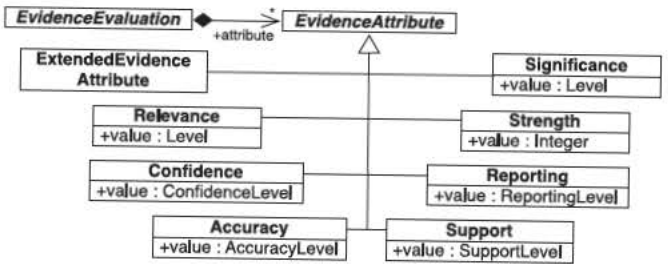

Fig. 15. Evidence attribute class diagram.

With the *Evidence Interpretation* class diagram (Fig. 16), someone could specify that a given report (evidence element) is a requirements specification (formal element), that the software verification results (evidence element) mean that 'actions have been taken for system verification' (formal assertion), and that a hazard log (evidence element) is scoped by the system or the component under consideration (formal element).

In our opinion, and as explained above for other classes, all the specialisations *of Evidence Interpretation* should be turned into associations, as their multiplicity is '1' and the classes do not seem to have further relevant properties. This would also imply that *Evidence Interpretation* usage could be supported by associations, without any new class. In addition, the specialisations seem to overlap with the classes of the Argumentation Metamodel (except *Is Scoped By* and *Is A*) and of the Formal Statements Section. This should be revised. Again, and as for other class diagrams, the possible way of using this class diagram is probably too broad. It should more clearly specify what evidence elements (e.g., *Exhibit*) can be interpreted.

The *Evidence Observation* class diagram (Fig. 17) clearly overlaps with other parts of SACM. For example, a conflict can be specified with the Argumentation Metamodel (see Section 3.9).

Regarding *Evidence Resolution* (Fig. 18) and its specialisations, we have found the same issues as in *Evidence Observation* regarding its overlap with argumentation. Some constraints might also be needed for the correct use of the specialisations. For example, *Resolves* should only be used for conflicts. Some descriptions could also be improved. The difference between *Refutes* and *Challenges* could be explained in more detail, and presenting *Negates* in SACM as an indirect refutation can be confusing.

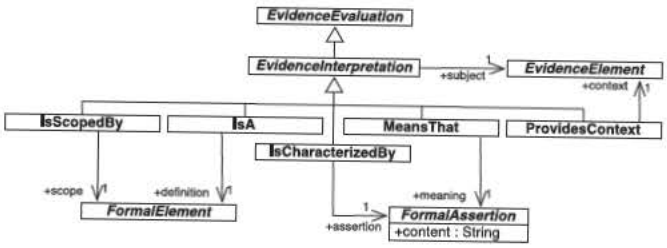

Fig. 16. Evidence interpretation class diagram.

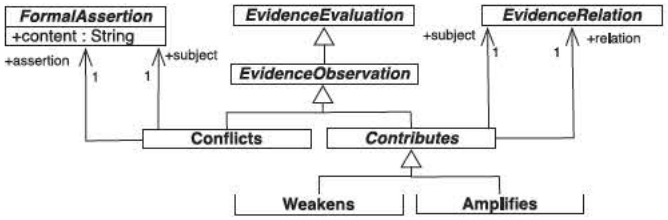

Fig. 17. Evidence observation class diagram.

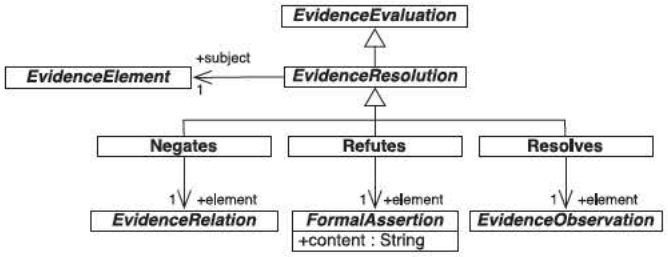

Fig. 18. Evidence resolution class diagram.

Finally, our main overall concern with *Evidence Evaluation* is that most of the information that can be specified with it falls into the scope of argumentation and not of the management of the artefacts that can be used as safety evidence per se. This should be revised for an adequate, more integrated usage of the Evidence Metamodel and the Argumentation Metamodel.

3.8. Administration

The Administration section of the Evidence Metamodel consists of the *Project*, *Project Elements*, and *Project Properties* class diagrams. This section describes the elements of the Evidence Metamodel that are involved in managing and exchanging evidence, and defines the root object of an Evidence Model: *Evidence Container*.

The *Project* class diagram (Fig. 19) shows the components of *Evidence Container*. An evidence container could, for instance, en compass all the information about the artefacts used as safety evidence for a system component: the information about the component assurance process. As explained below, the composition association between *Project Property* and *Project Element* is too broad. Some specialisations of *Project Element* do not seem to need *Project Property*, e.g. it does not seem necessary to indicate that an evidence request has a role in an organization. The inclusion of *Evidence Evaluation* in *Evidence Container* represents to us a clear overlap with the Argumentation Metamodel. It would also probably be more suitable that evidence evaluations belong to the evaluated evidence element, not to evidence containers.

With the *Project Elements* class diagram (Fig. 20), someone could specify that software verification results are created for showing that a system satisfies its requirements (project objective) and that this artefact results from a software verification activity (activity) in which a system verifier (person) participates. Project elements relate to the process followed for system assurance. Modelling of this process is relevant for e.g. safety critical systems [47].

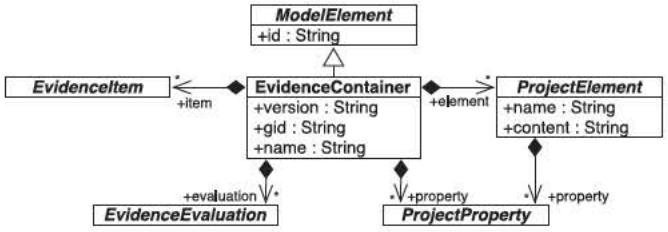

Fig. 19. Project class diagram.

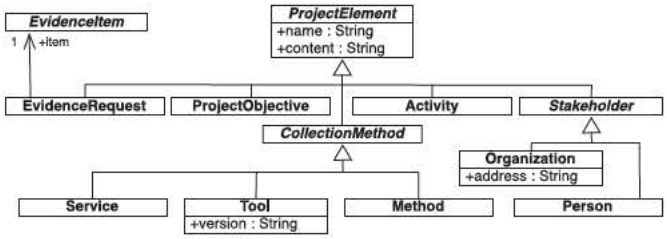

Fig. 20. Project element class diagram.

The main issues that we have found in this class diagram are not strictly in itself, but in its associations and relationships with others. More concretely, the notion of *Evidence Request* is unclear to us because *Evidence Item* is unclear too. We have mentioned the issue with *Project Property* in the previous paragraph. *Project Element* is also a specialisation of *Evidence Element* (see Fig. 4), which has many characteristics that do not seem to apply to *Project Element*. We do not clearly see why someone would need to specify for assurance purposes that e.g. a person takes care of a method. Further inconsistent or incoherent information such as 'the organization O has been created using the tool T' can be specified with SACM. These issues must be fixed by adding constraints or changing the specialisation hierarchies. We also wonder about the extent to which *Project Objective* and *Claim* (from the Argumentation Metamodel) overlap. Although the descrip tion of *Activity* indicates that it can be used to specify an executed action or an action to execute, no mechanism is provided to distinguish between these states. Finally, we do not understand why *Collection Method* includes 'Collection' in its name. Its specialisations, and the means used for safety evidence management in general, do not aim only at evidence collection, but could also aim at other actions such as artefact generation or verification (e.g., a software testing tool).

As mentioned above for other classes, most of the classes that specialise *Project Property* (Fig. 21) are not relevant for all the specialisations of *Project Element*, and *Requires Container*, *Depends On*, and *Satisfies* could be turned into associations ('1' multiplicity criterion). The scope of *Depends On* and of *Satisfies* seem too broad. More information is necessary about of types of the pairs of project elements that can be related via these concepts. It does not seem suitable to us to specify e.g. that a given tool satisfies a given person. SACM should further explicitly indicate which project properties could belong to an evidence container and which to a project element.

The proposed values for *Complies To*, *Container Consistency* and *Container Completeness* might not fit the current practices and criteria of an organization, as explained above for *Document Property*. These classes might also need constraints for their adequate usage (e.g., can an evidence container be complete if some contained document is not?) and should probably be modelled as attributes of *Evidence Container* (see discussion above for e.g. evidence attributes). Although the enumeration for *Complies To* is called *Standard Of Proof*, it does not refer to compliance with e.g. safety standards.

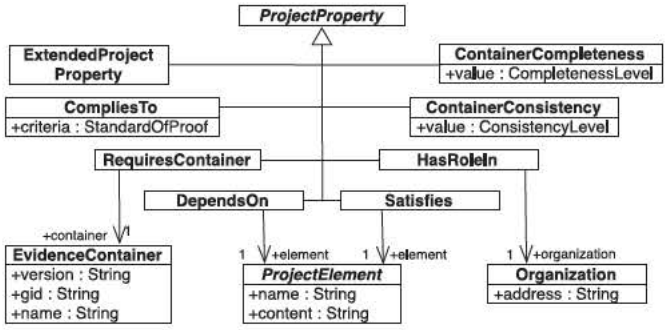

Fig. 21. Project property class diagram.

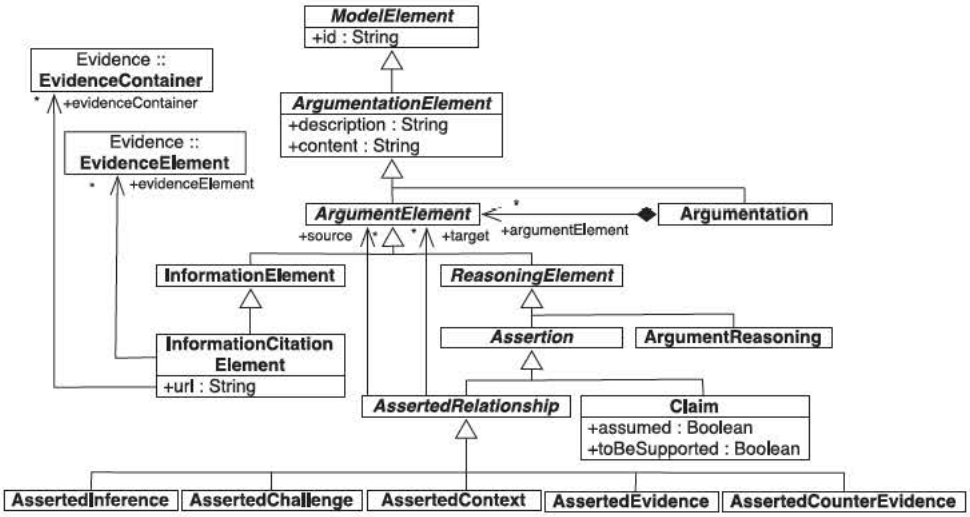

Fig. 22. Argumentation class diagram.

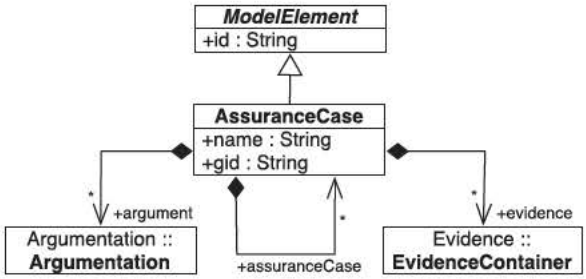

Fig. 23. Assurance case structure.

3.9. Argumentation

By argumentation, we refer to the elements of the Argumentation Metamodel, and more concretely to those related to safety evidence. This metamodel corresponds to the *Argumentation* class diagram (Fig. 22). In SACM, an assurance case (Fig. 23) consists of a set of *Argumentations* (from the Argumentation Metamodel) that correspond to its argument, and a set of *Evidence Containers* (from the Evidence Metamodel) that correspond to its evidence.

An evidence element (e.g., the software verification results) or evidence container (all the evidence information for a given software component) can be used in an *Argumentation* by means of *Information Citation Element*. This *Argumentation Element* enables the citation of a source of information that relates to a structured argument. The declaration of relationship is made by means of *Asserted Relationship*. This class is specialised into *Asserted Inference*, *Asserted Evidence*, *Asserted Counter Evidence*, *Asserted Context* and *Asserted Challenge*. The source or *Asserted Evidence* and *Asserted Counter Evidence* must be *Information Element*.

The main issue that we find in the Argumentation Metamodel is that the relation between *Evidence Element* and *Information Citation Element* implicitly means that any *Evidence Element* can be used as *Evidence*. As discussed in Section 3.1, we think that this is not correct, or at least that it is ambiguous. It is also hard to understand why some classes of the Evidence Metamodel and the Argumentation Metamodel seem to overlap. Three specialisations of *Asserted Relationship* used for claims (*Asserted Evidence*, *Asserted Counter Evidence*, *Asserted Inference*, and *Asserted Challenge*) seem to be very similar to classes of the Evidence Metamodel that can be used for *Formal Assertion* (e.g., *Supports* and *Challenges* in the Evidence Relation class diagram). As indicated above, all argumentation related aspects should be moved to the Argumentation Metamodel.

Another possible inconsistency between the Evidence Metamodel and the Argumentation Metamodel is the existence of a circular reference. *Claim* is referred to in the Formal Assertions class diagram as a class associated to *Referenced Claim*. However, we understand that the link between the Evidence Metamodel and the Argumentation Metamodel should only be via the use of evidence in an argument. We further doubt that *Referenced Claim* is a really necessary class, taking into account the conceptual modelling principles regarding what an entity type is or if two entity types represent the same concept [11]. Nonetheless, and as acknowledged for other SACM classes, this could correspond to a design or implementation decision whose rationale is not described in the standard. The Argumentation Metamodel also allows the use of *Asserted Evidence* and *Asserted Counter Evidence* for modelling evidentiary support of *Information Element* to more than one claim. We think that this can lead to weak or inadequate argumentation structures because some characteristics of a piece of evidence depend on single claims [48]. For example, the use of a hazard log as evidence could be appropriate for a claim such as "System hazards have been identified and recorded" but not for "All the possible system hazards have been mitigated or avoided". For the latter, there might still exist some unknown system hazards.

4. Discussion

This section presents improvement opportunities and extension possibilities in SACM, as a way to synthesise and summarise the analysis in Section 3. They represent recommendations so that the standard better supports safety evidence management. We also discuss analysis validity.

It must be noted that the current SACM version was released some months ago, and that it is common that OMG standards undergo revisions as improvement opportunities are found. In fact, there are already plans for a 2.0 version [49]. The analysis in the previous section and the discussion in this section must not be regarded only as weaknesses or important deficiencies in SACM, but as aspects that might be taken into account in future versions in order to make the standard more suitable for safety evidence management and to facilitate its application.

According to Martin [50], SACM 2.0 will aim to improve its support for IEC 15026 2 and for GSN, to harmonize some parts, to support patterns, templates, and structured expressions, and to improve SACM modularity and simplicity. These aspects complement those discussed below. Unlike Martin [50], we provide details about how to improve and extend SACM, and focus on safety evidence management. It must also be mentioned that we analysed SACM and listed the improvement

opportunities and extensions possibilities before being aware of [50], thus it has not influenced us.

As we have discussed in similar analyses [17], the improvement opportunities and extension possibilities that we propose are not unsubstantial or unnecessary. They can contribute to improving SACM and to making the standard more suitable for assurance in general and for safety evidence management in particular. For exam ple, we argue that a clearer definition of SACM would lead to a more effective standard. A simpler definition, in which e.g. overlaps are reduced, would probably also contribute to this objective.

4.1. Improvement opportunities

By improvement opportunities, we mainly refer to the existence of classes and associations that might need to be clarified or might not be necessary (e.g., they are redundant). These improvements would enable an easier understanding of SACM use and thus an easier use for safety evidence management. We consider that the main general improvement opportunities in SACM can be summarised as follows.

1) *Inclusion of more examples*. The area that probably requires further improvement in SACM is the lack of examples for the vast majority of classes and associations of the Evidence Metamodel. SACM annexes just provide a small Argumentation Model, we have not found Evidence Models in the literature, and the few examples in the Evidence Metamodel are too abstract in our opinion. Provision of examples will be essential to clarify the classes and associations, and thus to be able to determine the real need for removing or modifying them. The examples would also provide further guidance on how and when to use SACM elements.
2) *Clarification of the notion of evidence*. Section 3.1 has discussed the notion of evidence in SACM, indicating several possible issues. We consider that the main issue comes from using *Evidence Item* and *Evidence Element* for a variety of concepts that, to our understanding, do not correspond to evidence in general and to safety evidence in particular. An alternative term might be used. For example, the term Assurance Asset is used in OPENCOSS for classifying assurance information that contributes to developing confidence in the safe operation of a system, such as an artefact or the execution of an activity [51].
3) *Clarification of the notion of evidence assertion*. It is not comple tely clear what SACM means by *Evidence Assertion*, thus we argue that this concept needs to be better defined in the standard. To our understanding, it is an assertion about a piece of (safety) evidence that does not need to be justified and that does not require an argument. The corresponding artefact will suffice to show assertion validity. For the hazard log, a possible evidence assertion is "The hazard log includes system hazards". Furthermore, *Evidence Assertion* is used for aspects such as the properties of a document. We think that, as in other metamodels and in general, properties should simply be included in the classes (e.g., *Document*) as attributes, instead of implying that a property is something asserted about a document.
4) *Reduction in the number of classes and associations*. Based on conceptual modelling principles [11], several classes and associa tions of the Evidence Metamodel are probably not necessary. Basically, several classes could or even should be modelled as associations. A high number of elements in a metamodel also affects negatively its quality [12,15], making it more complex to under stand and maintain.
5) *Restructuring of classes and associations*. The restructuring of several SACM parts has been proposed in Section 3. Among them, we regard as especially important to carefully restructure the Formal Statements Section, since it introduces circular references in the metamodel. SACM would also benefit from a clearer separation of evidence concerns and argumentation concerns. Several classes of the Evidence Metamodel seem to fall into the scope of argumentation, as their use might be associated with an argument that justifies the information provided (e.g., *Reliability* of *Document*).
6) *Redundancy reduction*. Some kinds of information can be specified in several ways with SACM. This is a result of the overlaps that seem to exist between classes and associations. For example, it seems that *Supports* and *Asserted Evidence* can be used for the same purpose. Redundancy is a characteristic that should be avoided in a metamodel [12,15], thus overlaps between classes must be ad dressed.
7) *Reduction in the scope of some classes*. We have indicated that the scope of several classes (e.g., *Project Element*) is probably too broad because not all their specialisations have all their attributes and associations. This can be solved by specifying constraints or by restructuring the attributes and associations of the specialisations.
8) *Further justification of the need for the classes and associations*. In general, the need for many classes of the Evidence Metamodel, their purpose, and how to use them should be presented in more detail. Otherwise, it is difficult to determine if a class should be in the metamodel, or if an instance of the class should be included in a model. This might be a common issue in OMG standardisation efforts [52]. There is usually a rationale behind all the elements of a metamodel although it is not provided.
9) *Consistent concept definition*. Several key concepts are defined several times in SACM, in different parts of SACM document (e.g., *Evidence Item*, *Evidence Assertion*, *Claim*, and *Argument*). This can make SACM difficult to understand, especially when the definitions are inconsistent and considering the central role of some concepts for assurance case creation and safety evidence management. For example, one of the definitions of *Evidence Item* in SACM indicates that claims are evidence items, but others do not and only refer to e.g. physical elements. Problems with terminology and difficulties to agree upon it have been reported for other OMG standards [53].

It must be noted that most of the improvement opportunities above are not independent, but that relations exist between them. For instance, the inclusion of more examples might help to clarify the notions of evidence and evidence assertion.

Table 1 indicates which SACM aspects (subsections of Section 3) would benefit from each improvement opportunity. From a general perspective, the improvement opportunities that we have found most frequently, based on the number of SACM aspects, are the clarification of the notion of evidence and the further justification of the need for the classes and associations (seven aspects). Evidence Evaluation is the SACM aspect that can benefit from a highest number of improvement opportunities.

In addition to the above improvement opportunities, the following aspects would also represent an advantage for SACM:

- Further tool support, which might have allowed the identification of some of the issues mentioned in this paper and facilitate the identification of others; we are not aware of any tool that imple ments the Evidence Metamodel of SACM 1.0 or 1.1.
- Graphical representation of all the attributes and associations of the classes, so that a user can gain a deeper understanding of the scope of each class when checking the class diagrams.
- Text revision, including class, attribute, and association naming, for fixing some errors and helping a reader to more easily understand the standard.

4.2. Extension possibilities

By extension possibilities in SACM, we refer to the existence of concepts and relationships that are not represented in the standard but

Table 1
Improvement opportunities outline.

		SACM aspect								
		Notion of evidence	Evidence lifecycle	Evidence elements	Exhibit properties	Formal statements	Evidence properties	Evidence evaluation	Administration	Argumentation
Improvement opportunities	Inclusion of more examples			X	X	X	X	X	X	
	Clarification of the notion of evidence	X	X	X	X		X	X		X
	Clarification of the notion of evidence assertion			X	X	X	X	X		X
	Reduction in the number of classes and associations				X		X	X	X	
	Restructuring of classes and associations				X	X	X	X		
	Redundancy reduction		X				X	X		X
	Reduction in the scope of some classes	X	X	X			X	X	X	
	Further justification of the need for the classes and associations	X		X	X	X	X	X	X	
	Consistent concept definition	X	X	X	X			X		X

might be necessary for safety evidence management. These extensions would make SACM more suitable for this activity. We consider that the main extension opportunities relate to the following areas.

1) *Evidence Item and Evidence Assertion lifecycles*. We have indicated that evidence items and evidence assertions, as well as pieces of safety evidence, artefacts, and assurance information in general, can have more states than those currently represented in SACM. Therefore, the lifecycles proposed in SACM could be extended. This extension could also include the specification of preconditions for the lifecycle events.

2) *Evidence usage throughout its lifecycle*. We have explained that evidence lifecycle aspects such as the specification of the version of a document that is used in an activity are not adequately represented in SACM. Therefore, a wider, better support for evidence usage and lifecycle is an area that can drive some extensions in SACM.
3) *Evidence change impact analysis*. Pieces of safety evidence, as well as assurance cases, evolve during their lifecycle. The corresponding changes might impact other elements of an SACM Model, and actions might have to be taken so that a body of safety evidence remains valid. Change management and impact analysis are barely supported in SACM currently, and it might be useful to take them into account in future versions. Nonetheless, we acknowledge that evidence change impact analysis and assurance case evolution are areas that probably require further research before effectively defining common, standardised practices.
4) *Means for compliance with safety standards*. Last but not least, SACM does not explicitly support the specification of how a system complies with a safety standard. Although it could be done by means of, for instance, claims or formal assertions, we think that explicit support in the form of classes and associations for compliance modelling would be positive. This kind of approach has been proposed and discussed in [19].

4.3. Validity

We discuss the validity of our analysis according to the perspectives proposed by Wohlin et al. [54].

Construct validity is concerned with the relationship between a theory behind an investigation and its observation. Some issues that have been indicated might simply correspond to errors in the SACM document (e.g., the multiple definition of *Evidence Item*). In addition, no standard definition of safety evidence exists and we have adopted a definition [5] as basis for the analysis. Nonetheless, such a definition is based on a large scale systematic literature review [5] and has been later validated in surveys with practitioners [6,7]. This makes us confident in its validity. Having analysed SACM with real data from OPENCOSS partners might threat construct validity because data from other real projects might show different characteristics of safety evidence management practice.

Internal validity is concerned with the causal, observed relationship between a treatment and its results. Although we consider that we have been cautious and rigorous, three main characteristics of the analysis might have posed threats to internal validity. First, our background on model driven engineering and conceptual modelling (e.g., [17,19]), and thus our experience in creating, using, and analysing other metamodels and conceptual models, might have made us study SACM from a given perspective. This could have been the reason of, for instance, finding an issue in regarding *Document Property* as *Evidence Assertion* (Section 3.3). Second, most of the issues indicated are based on insights from OPENCOSS. We trust the insights from prior studies in OPENCOSS, since they have been large (high number of data points) and wide (heterogeneity of the sample), but cannot guarantee that other safety evidence management aspects did not need to have been taken into account. Third, there is a threat in the possibility of having misinterpreted SACM text and class diagrams. We consider that this threat is inherent to the analysis of any text, including the text of a modelling standard. Nonetheless, all the authors reviewed the results of the analysis to mitigate the threat, and when considered necessary, we discussed the results to agree upon a common interpretation of SACM text and class diagrams. Finally, there is an

inherent threat to internal validity due to the fact that we used a specific set of examples and real cases as basis for the analysis, including for the creation of SACM models. Different examples and cases might yield different analysis results. For example, other issues might be identified in SACM.

Conclusion validity is concerned with the relationship between a treatment and its outcome. In our opinion, the main threat is that the analysis performed is probably beyond SACM scope. SACM focuses on evidence for assurance cases, whereas safety evidence management is a wider activity. For example, most of the extension possibilities seem to be out of the current SACM scope, thus this is probably the reason why some concepts are not currently represented in the standard. Anyway, it is also true that SACM 1.1 includes classes and associations for managing evidence beyond their use in an assurance case (e.g., *Activity*). The threats to construct and internal validity must also be taken into account before drawing definite conclusions about SACM support for safety evidence management.

External validity is concerned with the generalization of the conclusions of an investigation. The analysis performed on SACM has focused on how it supports safety evidence management, thus it can be hard to generalise the findings to other assurance activities. Nonetheless, we believe that other activities can also benefit from most of the improvements and extensions proposed. Most of the aspects acknowledged for safety evidence also apply to any type of evidence (e.g., clarification of the notion of evidence), and evidence is the basis for other activities (e.g., argumentation). Many issues are inherent in SACM (e.g., unnecessary classes), regardless of its usage purpose. Regarding the threat of having used a reduced set of examples (e.g., hazard log and software verification results) for showing how SACM can be used and its limitations, we argue that the examples used are valid representatives of what safety evidence is and how it is managed.

5. Conclusion

Provision of assurance is essential for many critical systems, and the specification of structured assurance cases is a common practice in many domains. Several initiatives currently exist towards the definition of standard means for assurance case specification. Safety evidence must be further managed as part of the assurance process and assurance case specification. Therefore, standardisation efforts for assurance case specification must take safety evidence management into account.

This paper has analysed how SACM (Structured Assurance Case Metamodel) supports safety evidence management. To the best of our knowledge, the paper is the largest and deepest available SACM analysis. Past analyses have dealt with very few SACM elements and have not presented details about the insights provided. Further, the analysis is very valuable for those studying SACM adoption, as it indicates possible issues and potential needs to address.

We have found issues or possible issues in all SACM class diagrams. Among the nine main improvement opportunities identified, we think that the most important areas, and thus the ones that should be addressed sooner, are the provision of more examples, the clarification of the notions of evidence and of evidence assertion, and a thorough analysis of the scope and of the need for some classes, including their overlaps. These improvements would significantly contribute to clarifying SACM and how to use it. They would also impact the rest of the improvement opportunities because it would be easier to determine the extent to which they correspond to real issues in SACM and thus their importance. In addition, we have proposed four areas related to safety evidence management based on which SACM could be extended: evidence item and evidence assertion lifecycles, evidence usage throughout its lifecycle, evidence change impact analysis, and compliance with safety standards. Nonetheless, these aspects might be out of the scope of SACM if the standard aims to focus on evidence management for assurance case specification and thus does not need to support other activities (e.g., change impact analysis). Therefore, a reader should be cautious when analysing SACM suitability for evidence management according to the proposed extensions.

Addressing the improvement opportunities is especially important. It would reduce SACM complexity, which would in turn facilitate its understanding, promote its use, and help a reader to find further improvement opportunities and extension possibilities. Based on the analysis, our overall conclusion is that SACM can be suitable for safety evidence management in the scope of a safety case, as long as the use of SACM is focused on the Argumentation Metamodel to specify the evidence elements that support or relate to safety arguments and claims, such as the evidence asserted for them. Few issues have been identified in the Argumentation Metamodel, most of them correspond to overlaps and inconsistencies with the Evidence Metamodel, and we think that these issues can be easily addressed. Nonetheless, the use of SACM should be tailored, selecting the classes from the Evidence Metamodel that are really necessary and deciding upon how to use them. For example, the use of all the SACM evidence elements as evidence in a safety case should be constrained because some of these elements (e.g., evidence assertions) should not be regarded as artefacts that can support a safety claim. It is very important that a user is aware of the possible limitations in SACM usage. For safety evidence management activities beyond the scope of a safety case, we argue that SACM is not the best alternative and that other models better support the whole safety evidence lifecycle (e.g., [20,51]). We also consider that provision of evidence in general and in SACM in particular should be artefact centred. This could be achieved by clarifying and reducing the scope of most of the content of the Evidence Metamodel.

Although several issues, improvement opportunities, and extension possibilities have been discussed, SACM provides a great basis towards standardizing and improving assurance case practices in general and safety evidence management practices in particular. Before its creation, most of the available knowledge was spread among different sources, e.g. literature on GSN, literature on CAE, and different safety assurance metamodels (see Section 2). Some insights were not even available, e.g. a classification of evidence events and a unified set of properties to evaluate pieces of evidence in the form of a document. There has also been a clear progress from the Beta versions of SACM to the 1.0 version, and from the 1.0 version to the 1.1 one, thus we expect also progress and improvements in the upcoming versions. It must also be noted that SACM needs to be generic and support general assurance and assurance case specification practices. SACM is not targeted only at safety evidence management, and it should not be targeted at project or company specific practices.

The analysis presented focuses on SACM 1.1. Although SACM might evolve in the future as OMG standards typically do, SACM 1.1 will be the version to use of the standard until a new one is published. The analysis points out aspects that could be addressed not only in future versions but also when using the current one, e.g. reducing the number of classes to use to avoid overlaps. Many insights from the analysis are also valuable to people interested in creating a metamodel or analysing other metamodels, as we indicate general modelling aspects to take into account, and we have presented information about how safety evidence management should be performed in order to explain the analysis. This information can be useful to anyone interested in safety evidence management. Finally, it is always necessary to analyse the current version of a given standard to provide suggestions for future versions, as it has been done in the past for e.g. BPMN and UML.

As future work, we plan to continue developing new approaches for improving safety evidence management practices. We are currently working on the definition of new means for automating safety evidence collection from different engineering tools and for safety evidence traceability. We would also like to perform a similar analysis to the OMG Dependability Assurance Framework [55] once it is released, and perform a detailed comparison of SACM with the metamodels created

in OPENCOSS [51] and with the rest of standardization efforts for assurance cases. Evidence requirements of specific safety standards and their comparison (as in e.g. [56]) could also be used as basis for analysing SACM, and the extent to which SACM models help understand assurance information could be studied, similarly to recent experiments on safety compliance needs [58]. Finally, and as for other OMG standards, further analyses can be performed on SACM in relation to, for instance, its ontological foundations (as for BPMN [15]) and its complexity (as for UML [16]).

Acknowledgement

The research leading to this paper has received funding from the FP7 Programme under the Grant agreement no. 289011 (OPENCOSS), from the H2020 ECSEL Programme under the Grant agreement no. 692474 (AMASS), and from Spain's MINECO ref. PCIN 2015 262. We thank the people with whom we have discussed SACM content and suitability, especially Katrina Attwood, Philippa Conmy, Luke Emmet, Huascar Espinoza, Tim Kelly, Bob Martin, Sunil Nair, Rajwinder Kaur Panesar Walawege, and Alejandra Ruiz. We are also grateful to the reviewers of the paper for the useful improvement suggestions.

Appendix A. Examples of SACM models

This appendix includes examples of SACM models in three different ways: UML object diagrams (Figs. 24 and 25), a screenshot of a basic SACM editor created with the Eclipse Modeling Framework [57] (Fig. 26), and a XML file corresponding to the output data of the editor. The main container element of the model shown in Fig. 25 ('System Evidence' evidence container from Fig. 24 (a)) is not included in the figure for simplicity and readability. The content of the models corresponds to the running examples used in Section 3. Models with real project data were also created in the scope of the OPENCOSS project. However, these models cannot be shared for confidentiality reasons.

The content of the XML file is as follows.

```
<?xml version "1.0" encoding "UTF 8"?>
<SACM:AssuranceCase xmi:version "2.0" name "System Assurance Case">
  <argument id "System Argumentation">
    <argumentElement xsi type "Argumentation:Claim" id "Safety Requirements Closure" description "All safety
  requirements have been closed"/>
    <argumentElement xsi type "Argumentation:Claim" id "Passed Reviews" description "All the reviews of the
  software system have been passed"/>
    <argumentElement xsi type "Argumentation:Claim" id "Hazards Identified" description "System hazards have
  been identified and recorded"/>
    <argumentElement xsi type "Argumentation:Claim" id "Hazard Absence"/>
    <argumentElement xsi type "Argumentation:AssertedEvidence" id "HL SRC" source "//@argument.0/@
  argumentElement.5" target "//@argument.0/@argumentElement.0"/>
    <argumentElement xsi type "Argumentation:InformationCitationElement" id "Hazard Log Evidence"
  evidenceElement "//@evidence.0/@item.2"/>
    <argumentElement xsi type "Argumentation:AssertedCounterEvidence" id "HL HA" source "//@argument.0/@
  argumentElement.5" target "//@argument.0/@argumentElement.3"/>
    <argumentElement xsi type "Argumentation:Claim" id "Hazard Mitigation or Avoidance" description "All the
  identified hazards have been mitigated or avoided"/>
    <argumentElement xsi type "Argumentation:AssertedEvidence" id "HL HI" source "//@argument.0/@
  argumentElement.5" target "//@argument.0/@argumentElement.2"/>
  </argument>
  <evidence name "System Evidence Container">
    <evaluation xsi type "Evidence:Challenges" id "HL HMA" assertion "//@evidence.0/@item.17" subject "//@
  evidence.0/@item.2">
      <attribute xsi type "Evidence:Confidence" value "reportedAsFact"/>
    </evaluation>
    <evaluation xsi type "Evidence:IsA" id "RS" subject "//@evidence.0/@item.4" definition "//@evidence.0/@
  item.14"/>
    <evaluation xsi type "Evidence:MeansThat" id "SVR" subject "//@evidence.1/@item.0" meaning "//@evi
  dence.0/@item.18"/>
    <evaluation xsi type "Evidence:IsScopedBy" id "HL SC" subject "//@evidence.0/@item.2"/>
    <item xsi type "Evidence:EvidenceGroup" name "Safety Analysis Artefacts" element "//@evidence.0/@item.1
  //@evidence.0/@item.3 //@evidence.0/@item.2"/>
    <item xsi type "Evidence:Document" name "FTA"/>
    <item xsi type "Evidence:Document" id "DOC HL 001" name "Hazard Log" url "" title "System Hazard Log"
  citation "System Manufacturer. Hazard Log of the System. 2016">
      <custody xsi type "Evidence:CareOf" custodian "//@evidence.0/@element.0"/>
      <custody xsi type "Evidence:UsingProcess" method "//@evidence.0/@element.1"/>
      <provenance xsi type "Evidence:OwnedBy" owner "//@evidence.0/@element.0">
        <timing xsi type "Evidence:StartTime"/>
      </provenance>
```

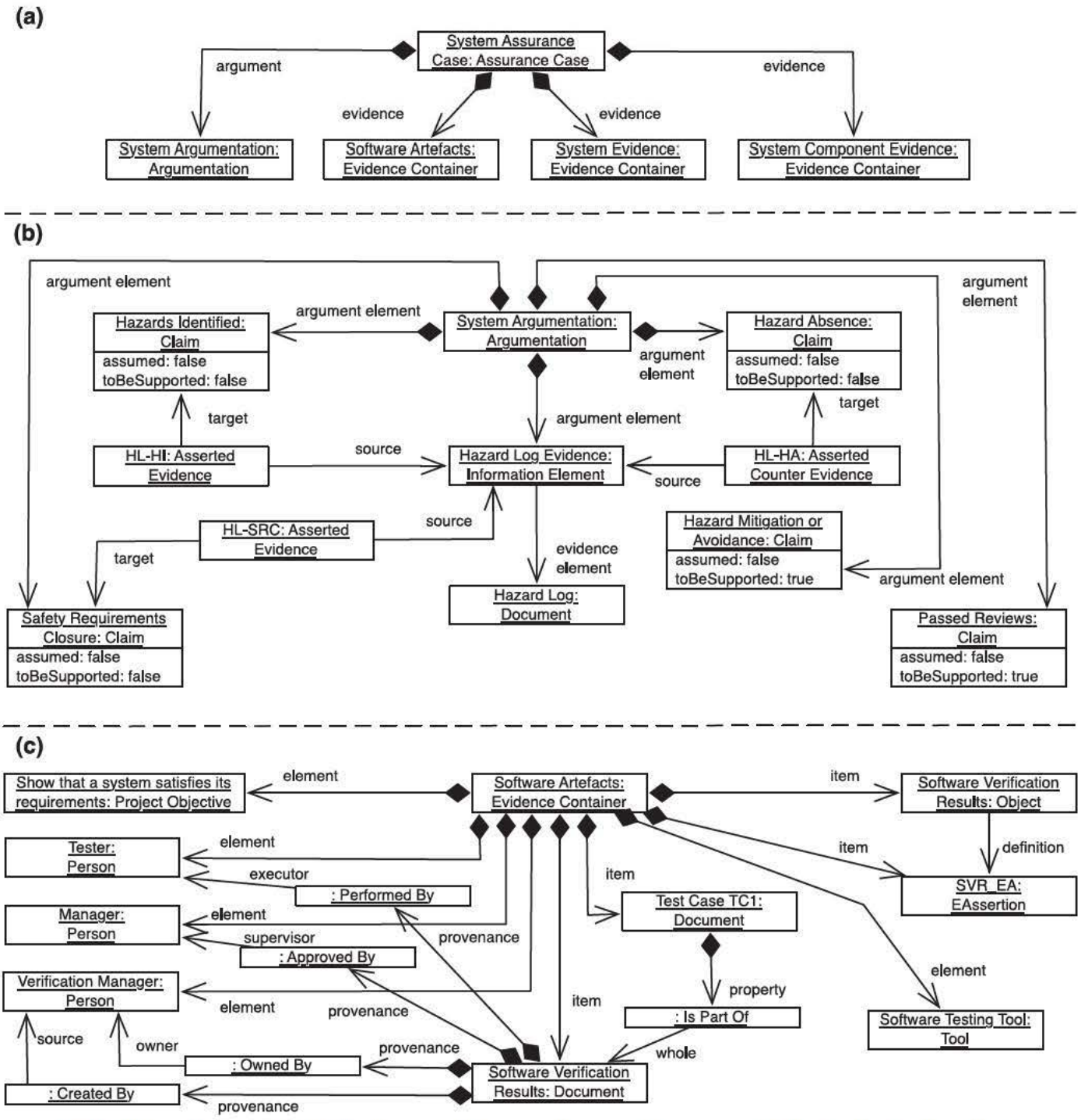

Fig. 24. SACM model in the form of a UML object diagram: (a) assurance case main structure; (b) argumentation; (c) evidence container for software artefacts.

```
    <property xsi type "Evidence:HasVersion" version "0.5"/>
    <property xsi type "Evidence:IsBasedOn" source "//@evidence.0/@item.6"/>
    <property xsi type "Evidence:HasElectronicSource" id "" source "Spreadsheet"/>
    <property xsi type "Evidence:Completeness" status "incomplete"/>
  </item>
  <item xsi type "Evidence:Document" name "FMEA"/>
  <item xsi type "Evidence:Document" name "Requirements Specification"/>
  <item xsi type "Evidence:Document" name "Design Specification"/>
  <item xsi type "Evidence:Document" name "Safety Plan"/>
  <item xsi type "Evidence:Exhibit" name "Architecture Model">
    <property xsi type "Evidence:HasElectronicSource" source "SysML diagram"/>
```

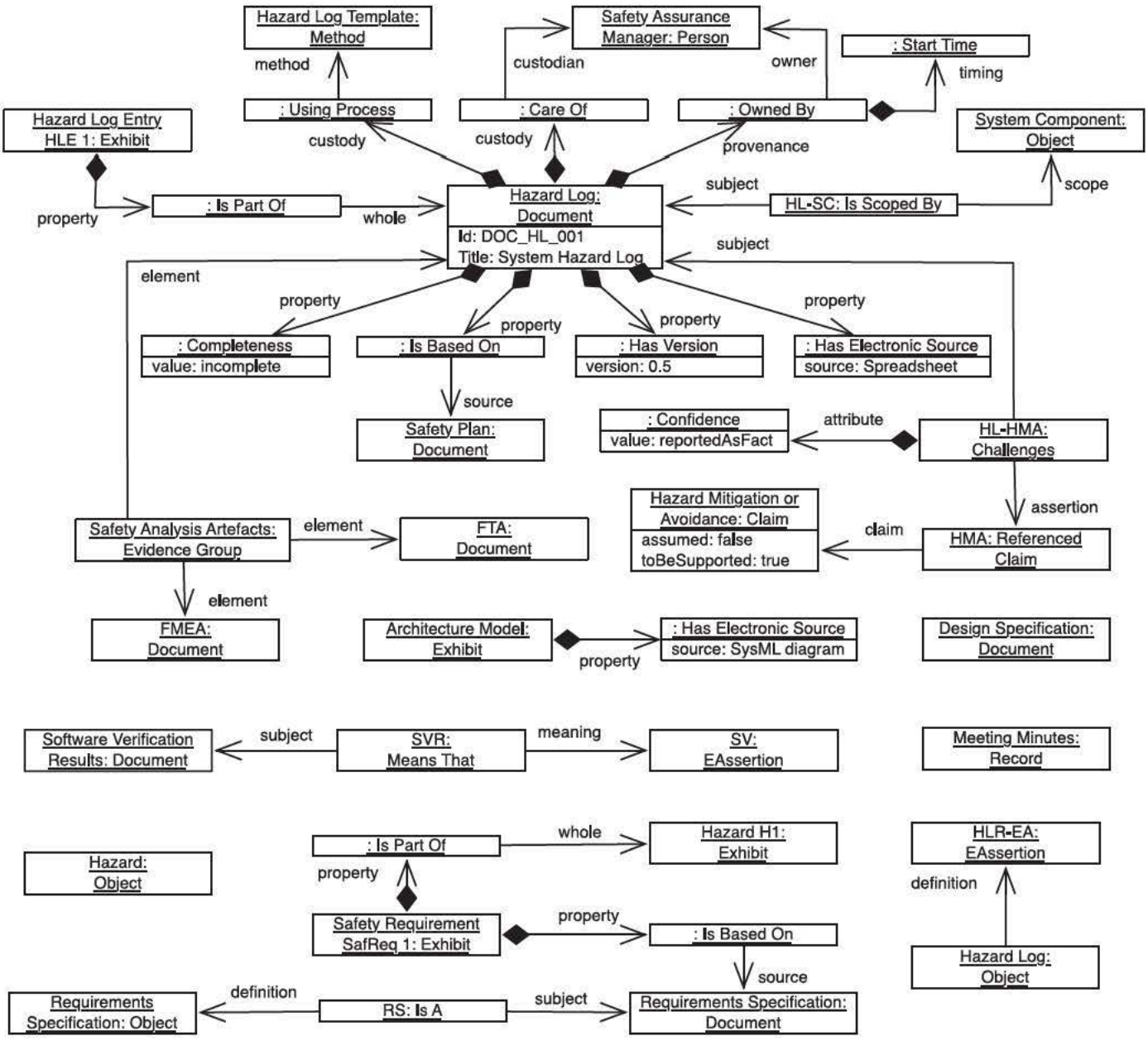

Fig. 25. SACM model in the form of a UML object diagram: evidence container for system evidence.

```
  </item>
  <item xsi type "Evidence:Exhibit" name "Hazard H1"/>
  <item xsi type "Evidence:Exhibit" name "Hazard Log Entry HLE1">
  <property xsi type "Evidence:IsPartOf" id "" whole "//@evidence.0/@item.2"/>
  </item>
  <item xsi type "Evidence:Exhibit" name "Safety Requirement SafReq1">
    <property xsi type "Evidence:IsPartOf" whole "//@evidence.0/@item.4"/>
    <property xsi type "Evidence:IsBasedOn" source "//@evidence.0/@item.8"/>
  </item>
  <item xsi type "Evidence:Record" name "Meeting Minutes"/>
  <item xsi type "Evidence:Object" name "Hazard"/>
  <item xsi type "Evidence:Object" id "" name "Hazard Log" concept "Hazard Log" definition "//@evidence.0/@
item.16"/>
  <item xsi type "Evidence:Object" id "" name "Requirements Specification"/>
  <item xsi type "Evidence:Object" name "System Component"/>
  <item xsi type "Evidence:EAssertion" id "HLR EA" content "A hazard log id the document in which all the safety
management activities, hazards identified, decisions made, and solutions adopted, are recorded or referenced"/>
  <item xsi type "Evidence:ReferencedClaim" id "HMA" content "" claim "//@argument.0/@argumentElement.7"/
```

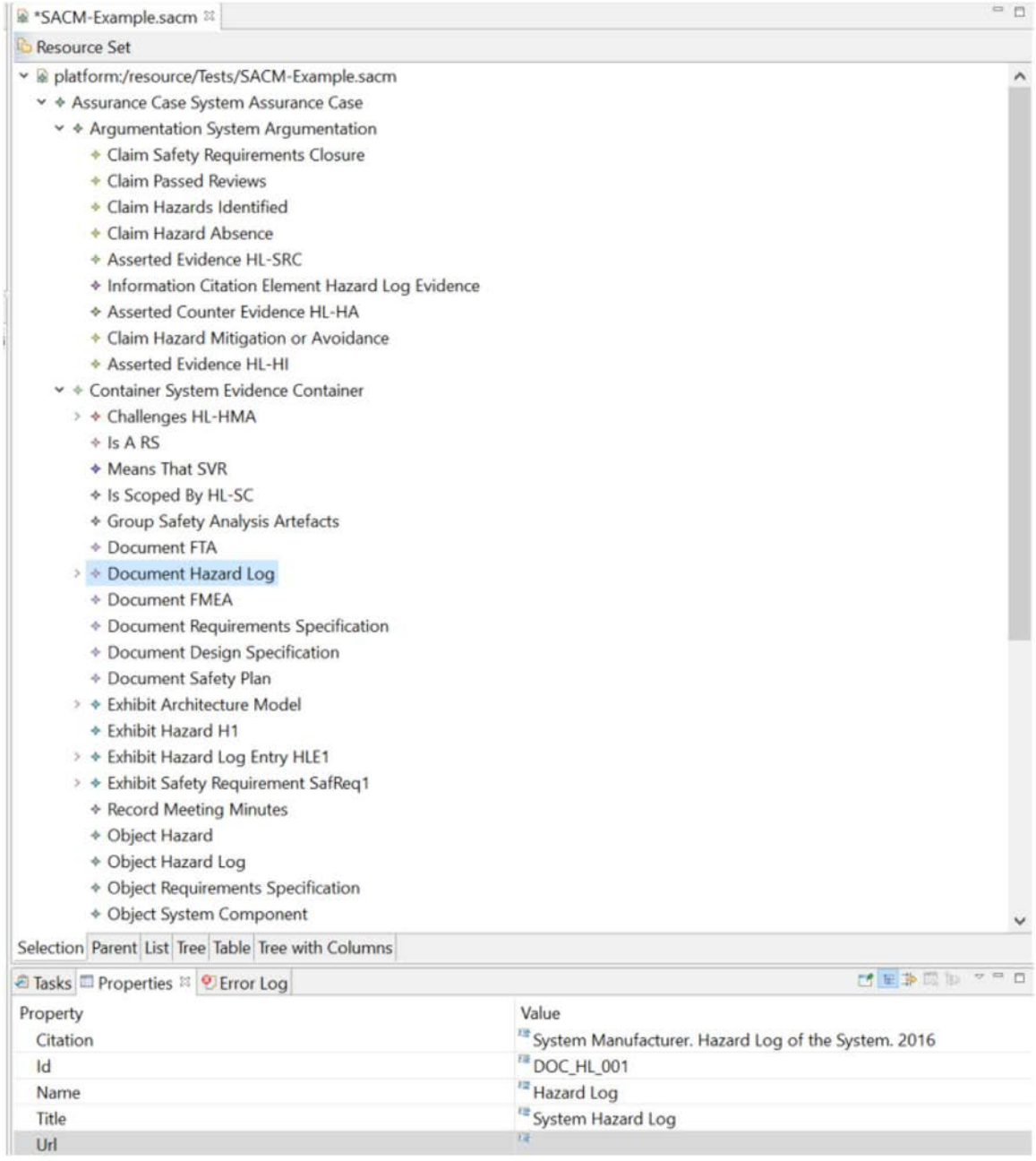

Fig. 26. Screenshot of a basic editor for creating SACM models.

```
>
  <item xsi type "Evidence:EAssertion" id "SV" content "Actions have been taken for system verification"/>
  <element xsi type "Evidence:Person" id "" name "Safety Assurance Manager"/>
  <element xsi type "Evidence:Method" name "Hazard Log Template"/>
</evidence>
<evidence name "Software Artefacts">
  <item xsi type "Evidence:Document" name "Software Verification Results" title "">
    <provenance xsi type "Evidence:OwnedBy" owner "//@evidence.1/@element.0"/>
    <provenance xsi type "Evidence:CreatedBy" source "//@evidence.1/@element.0"/>
    <provenance xsi type "Evidence:PerformedBy" id "" executor "//@evidence.1/@element.1"/>
    <provenance xsi type "Evidence:ApprovedBy" supervisor "//@evidence.1/@element.2"/>
  </item>
  <item xsi type "Evidence:Document" name "Test Case TC1">
    <property xsi type "Evidence:IsPartOf" whole "//@evidence.1/@item.0"/>
  </item>
  <item xsi type "Evidence:Object" name "Software Verification Results" definition "//@evidence.1/@item.3"/>
```

```
    <item xsi type "Evidence:EAssertion" id "SVR EA" content "Software verification results indicate the pro
  cedures that passed or failed for each software review, analysis, and test, including coverage analyses and
  traceability analyses"/>
    <element xsi type "Evidence:Person" name "Verification Manager"/>
    <element xsi type "Evidence:Person" name "Tester"/>
    <element xsi type "Evidence:Person" name "Manager"/>
    <element xsi type "Evidence:ProjectObjective" name "Show that a system satisfies its requirements"/>
    <element xsi type "Evidence:Tool" name "Software Testing Tool"/>
  </evidence>
  <evidence name "System Component Evidence"/>
</SACM:AssuranceCase>
```

References

[1] OMG, Structured Assurance Case Metamodel (SACM), version 1.1, ⟨http://www.omg.org/spec/SACM/1.1/⟩, 2015 (accessed 26.09.16), 2016.

[2] Astah, Astah GSN editor, ⟨http://astah.net/editions/gsn⟩, 2015 (accessed 26.09.16), 2016.

[3] OpenCert, ⟨https://www.polarsys.org/proposals/opencert⟩ (accessed 26.09.16), 2016.

[4] R. Hawkins, I. Habli, T. Kelly, J. McDermid, Assurance cases and prescriptive software safety certification: a comparative study, Saf. Sci. 59 (2013) 55 71.

[5] S. Nair, J.L. de la Vara, M. Sabetzadeh, L. Briand, An extended systematic literature review on provision of evidence for safety certification, Inf. Softw. Technol. 56 (7) (2014) 689 717.

[6] S. Nair, J.L. de la Vara, M. Sabetzadeh, D. Falessi, Evidence management for compliance of critical systems with safety standards: a survey on the state of practice, Inf. Softw. Technol. 60 (2015) 1 15.

[7] JJ.L. de la Vara, M. Borg, K. Wnuk, L. Moonen, An industrial survey of safety evidence change impact analysis practice, IEEE Trans Softw Eng, in press, 2016, http://dx.doi.org/10.1109/TSE.2016.2553032

[8] R. Alexander, T. Kelly, B. In: Gorry, Safety lifecycle activities for autonomous systems development, in: Proceedings of the 5th SEAS DTC Technical Conference, 2010.

[9] J.L. In: de la Vara, Current and necessary insights into SACM: an analysis based on past publications, in: Proceedings of the 7th International Workshop on Requirements Engineering and Law (RELAW 2014), pp. 10 13.

[10] OPENCOSS, D6.2 Detailed Requirements for Evidence Management of the OPENCOSS Platform, ⟨http://www.opencoss-project.eu/node/7⟩, 2012 (accessed 26.09.16), 2016.

[11] A. Olivé, Conceptual Modeling of Information Systems, Springer, Heidelberg, Germany, 2007.

[12] J. Becker, M. Rosemann, C. von Uthmann, Guidelines of business process modeling, in: W. van der Aalst, J. Desel, A. Oberweis (Eds.), Business Process Management LNCS, Springer, Heidelberg, 1806, pp. 39 40.

[13] OMG, Business Process Model and Notation (BPMN), ⟨http://www.bpmn.org/⟩ (accessed 26.09.16), 2016.

[14] OMG, Unified Modeling Language (UML), ⟨http://uml.org/⟩ (accessed 26.09.16), 2016.

[15] J. Recker, Evaluations of Process Modeling Grammars Ontological, Qualitative and Quantitative Analyses Using the Example of BPMN, Springer, Heidelberg, Germany, 2011.

[16] K. Siau, Q. Cao, Unified modeling language: a complexity analysis, J. Database Manag. 12 (1) (2001) 26 34.

[17] G. Génova, J. Llorens, A. Fraga, Metamodeling generalization and other directed relationships in UML, Inf. Softw. Technol. 56 (7) (2014) 718 726.

[18] D.J. Rainhardt , J.C. Knight , J. Rowanhill, Current Practices in Constructing and Evaluating Assurance Cases with Applications to Aviation, NASA Technical Report, 2015.

[19] J.L. de la Vara, A. Ruiz, K. Attwood, H. Espinoza, R.K. Panesar-Walawege, A. Lopez, I. del Rio, I., T. Kelly, Model-based specification of safety compliance needs: a holistic generic metamodel, Inf. Softw. Technol, 72, 2016, pp. 16 30.

[20] S. Nair, J.L. de la Vara, A. Melzi, G. Tagliaferri, L. de-la-Beaujardiere, F. Belmonte, Safety evidence traceability: problem analysis and model [- LNCS 8396], in: C. Salinesi, I. van de Weerd (Eds.), REFSQ, Springer, Heidelberg, 2014, pp. 309 324.

[21] N. Mansourov, D. Campara, System Assurance: Beyond Detecting Vulnerabilities, Morgan Kaufmann, Burlinton, MA, USA, 2011.

[22] J.L. de la Vara, H. In: Espinoza, Dealing with software model quality in practice: experience in a research project, in: Proceedings of the 1st International Workshop on Quality and Measurement of Software Model-Driven Developments (QUAMES), 2013, pp. 396 405.

[23] J.L. de la Vara, S. Nair, E. Verhulst, J. Studzizba, P. Pepek, J. Lambourg, M. Sabetzadeh, M. Towards, A model-based evolutionary chain of evidence for compliance with safety standards [Workshops LNCS 7613], in: F. Ortmeier, P. Daniel (Eds.), SAFECOMP, Springer, Heildelberg, 2012, pp. 64 78.

[24] L. Sun, Establishing Confidence in Safety Assessment Evidence, University of York, York, UK, 2013.

[25] OPENCOSS, D4.1 Baseline for the Common Certification Language, ⟨http://www.opencoss-project.eu/node/7⟩ (accessed 26.09.16), 2016.

[26] OPENCOSS, D6.1 Baseline for the Evidence Management Needs of the OPENCOSS platform, ⟨http://www.opencoss-project.eu/node/7⟩ (accessed 26.09.16), 2016.

[27] H. Li, J. Wu, C. Yuan, Y. Luo, M. van den Brand, L. In: Engelen, A Systematic Approach for safety evidence collection in the safety-critical domain, in: Proceedings of the 9th Annual IEEE International Systems Conference (SysCon), 2015, pp. 194 199.

[28] J.B. Goodenough, C.B. Weinstock, A.Z. Klein, Eliminative Argumentation: A Basis for Arguing System Properties. Software Engineering Institute, Technical Report, CMU/SEI-2015-TR-005, 2015.

[29] J. Rushby, Understanding and Evaluating Assurance Cases, SRI International, Technical Report SRI-CSL-15-01, 2015.

[30] R.K. Panesar-Walawege, M. Sabetzadeh, L. Briand, Supporting the verification of compliance to safety standards via model-driven engineering: approach, tool-support and empirical validation, Inf. Softw. Technol. 55 (5) (2013) 836 864.

[31] Goal Structuring Notation Working Group, GSN Community Standard version 1, ⟨http://www.goalstructuringnotation.info/documents/GSN Standard.pdf⟩, 2011 (accessed 26.09.16), 2016.

[32] Open Group, Dependability through Assuredness (O-DA) Framework, 2013.

[33] ISO, ISO/IEC 15026-2: Systems and Software Engineering Systems and Software Assurance Part 2: Assurance case, 2011.

[34] IEC: EN 62741 Guide to the Demonstration of Dependability Requirements The Dependability Case, 2015.

[35] L. Cyra, J. Górski, SCF a framework supporting achieving and assessing conformity with standards, Comput. Stand. Interfaces 33 (1) (2011) 80 95.

[36] EN, CENELEC 50129: Railway applications. Communication, Signalling and Processing Systems. Safety Related Electronic Systems for Signalling, 2003.

[37] DO-178C, RTCA: Software Considerations in Airborne Systems and Equipment Certification, 2012.

[38] Oxford Dictionaries, Evidence ⟨http://www.oxforddictionaries.com/definition/english/evidence⟩ (accessed 26.09.16), 2016.

[39] D. Jackson, M. Thomas, L.I. Millet, Software for Dependable Systems: Sufficient Evidence? The National Academic Press, 2007.

[40] D. Falessi, M. Sabetzadeh, L. Briand, E. Turella, T. Coq, R.K. Panesar-Walawege, Planning for safety standards compliance: a model-based tool-supported approach, IEEE Softw. 29 (3) (2012) 64 70.

[41] OPENCOSS, D6.3 Specification of the Evidence Management Service Infrastructure, ⟨http://www.opencoss-project.eu/node/7⟩ (accessed 26.09.16), 2016.

[42] G. Regan, F. McCaffery, K. McDaid, D. Flood, Medical device standards' requirements for traceability during the software development lifecycle and implementation of a traceability assessment model, Comput. Stand. Interfaces 36 (1) (2013) 3 9.

[43] OMG, Semantics of Business Vocabulary and Business Rules (SBVR), ⟨http://www.omg.org/spec/SBVR/1.0/⟩ (accessed 26.09.16), 2016.

[44] F. Jouault, I. Kurtev, Transforming Models with ATL [Workshops LNCS 3844], in: J.-M. Bruel (Ed.)Models, Springer, Heidelberg, 2005, pp. 128 138.

[45] P. Mäder, O. Gotel, Towards automated traceability maintenance, J. Syst. Softw. 85 (10) (2012) 2205 2227.

[46] J.P. Nytun, C.S. Jensen, Modeling and testing legacy data consistency requirements, in: P. Stevens, J. Whittle, G. Booch (Eds.), UML 2003 LNCS 2863, Springer, Heidelberg, 2003, pp. 341 355.

[47] C. Ayora, V. Torres, J.L. de la Vara, V. Pelechano, Variability management in process families through change patterns, Inf. Softw. Technol. 74 (2016) 86 104.

[48] S. Nair, N. Walkinshaw, T. Kelly, J.L. In: de la Vara, An evidential reasoning approach for assessing confidence in safety evidence, in: Proceedings of the 26th IEEE International Symposium on Software Reliability Engineering (ISSRE), 2015, pp. 541 552.

[49] OMG, System Assurance Task Force Agenda, ⟨http://sysa.omg.org⟩ (accessed 26.09.16), 2016.

[50] R.A. Martin, Building a Software Assurance Road-map and Using It Effectively, ⟨http://www.asq509.org/ht/a/GetDocumentAction/i/92786⟩ (accessed 26.09.16), 2016.

[51] OPENCOSS, D4.4 Common Certification Language: Conceptual Model, ⟨http://www.opencoss-project.eu/node/7⟩ (accessed 26.09.16), 2016.

[52] B.V. Selic, On the semantic foundations of standard UML 2.0, formal methods for the design of real-time systems, in: M. Bernardo, F. Corradini (Eds.), SFM-RT 2004

LNCS 3185, Springer, Heidelberg, 2004, pp. 181 199.
J. M.E. Morales-Trujillo, H. Oktaba, M. Piattini, The making of an OMG standard, Comput. Stand. Interfaces 42 (2015) 84 94.
K. C. Wohlin, P. Runeson, M. Höst, M.C. Ohlsson, B. Regnell, Experimentation in Software Engineering, 2nd ed., Springer, Heidelberg, 2012.
L. OMG, Dependability Assurance Framework for Safety-Sensitive Consumer Devices (DAF), ⟨http://www.omg.org/hot-topics/cdss.htm⟩ (accessed 26.09.16), 2016.
M. W. Youn, B.J. Yi, Software and hardware certification of safety-critical avionic systems: a comparison study, Comput. Stand. Interfaces 36 (6) (2014) 889 898.
[57] Eclipse Modeling Framework, ⟨https://eclipse.org/modeling/emf/⟩ (accessed 26. 09.16), 2016.
[58] J.L. de la Vara, B. Marín, G. Giachetti, C. Ayora, Do models improve the understanding of safety compliance needs? Insights from a pilot experiment, in: 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM 2016), pp. 32:1 32:6.